AN APPLICATION OF HEURISTICS IN MANAGEMENT DATA PROCESSING; THE SCHEDULING PROBLEM

HENRY J. DAVIS

AN APPLICATION OF HEURISTICS IN MANAGEMENT DATA PROCESSING; THE SCHEDULING PROBLEM

****

Henry J. Davis

AN APPLICATION OF HEURISTICS IN MANAGEMENT DATA PROCESSING; THE SCHEDULING PROBLEM

By

Henry J. Davis

Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS

United States Naval Postgraduate School
Monterey, California

1962

AN APPLICATION OF HEURISTICS IN MANAGEMENT DATA PROCESSING, THE SCHEDULING PROBLEM

by

Henry J. Davis

This work is accepted as fulfilling
the thesis requirements for the degree of
MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS
from the
United States Naval Postgraduate School

ABSTRACT

The problem of scheduling is one of assigning sets of given parameters to a time-table fulfilling certain constraints on assignments and avoiding conflicts between parameters. Large complex schedules that are to contain "desirable" features can be constructed by electronic data processing methods for symbol manipulation utilizing heuristics to reduce the search space.

Heuristic programming is an attempt to model human problem solving methods where steps toward the solution are not precisely defined. The mean-ends analysis of a problem in the General Problem Solver of Newell, Simon, and Shaw, as an application of heuristics to the scheduling problem, is given.

A computer program for the scheduling of classes at the U. S. Naval Postgraduate School is described. The method employs Boolean variables and expressions to describe attributes or characteristics of the parameters. Also employed is the principle of backtracking and a heuristic measure of complexity.

The writer wishes to express his appreciation for the assistance and encouragement given him by Professor M. L. Cotton of the U. S. Naval Postgraduate School in this investigation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. Introduction

Scheduling, as used in this report, is defined as the construction of a time-table or array, by assigning sets of given parameters to elements of the array. Time is to be alloted for specific combinations of parameters to join together for a given purpose. No parameter is to be scheduled to more than one function at one unit of time. There may be many separate sets of parameters scheduled to one unit of time if the parameters are disjoint.

There can be different classes of parameters to be scheduled in one problem, each class having its own specifications or constraints on scheduling. The constraints on assignment to the schedule may be given for a parameter class, or between classes, or between parameters and units of time.

Each functional set of parameters required to be scheduled together is called a unit of assignment. The assignment unit may consist of none or many elements of a parameter class, but the parameters essential to the function of the unit must be specified. Not all classes need to be specified in a unit; the problem may include the selection of an element from a parameter class from those that are available for assignment with the unit.

Scheduling, in accordance with this definition, has a number of management applications; the most obvious being that of scheduling people, as individuals or groups of individuals, to meet together with other groups at one time for a given function, and to meet with other groups at other times. One parameter class in this case would be people; the parameter space would be all sets of people to be scheduled. A parameter element would be one subset of people that are to be assigned together as a unit. Other parameter classes might be meeting places, or equipments required for the meetings.

Other examples are the scheduling of transportation facilities, scheduling of items for assembly line production, or the scheduling of equipment at a number of construction sites. In each case, the unit of assignment is known from the functional requirements, and the problem is to establish, by some criteria of employment, a schedule of the units to satisfy requirements.

Enough will be known about the problem such that an acceptable solution can be detected. An acceptable solution is one that satisfies all constraints; hence any acceptable solution is an optimum solution. There may be many optimum solutions possible; the goal is to obtain one optimum solution.

In a complex scheduling problem, particularly if the problem is recurrent, the methods of electronic data processing are particularly appropriate for the accomplishment of a schedule. High speed computers can perform the complex symbol manipulating properties of a schedule plan in a small fraction of the time required to do it manually.

The purpose of this report is to present a heuristic method for the construction of a schedule with a digital computer. A computer program for the heuristic solution of a school class scheduling problem is given in section four.

2. Methods of attacking the problem.

The feasibility of three approaches to the problem have been considered; methods of linear programming, search, and heuristics. This section contains a description of these methods as applied to the scheduling problem, and the reasons for the selection of the heuristic approach.

2.1 Linear programming.

The scheduling of parameters into a time array according to specified criteria of assignment is a variation of the so-called "assignment" problem of linear programming. In the assignment problem, it is assumed that, for example, measures of performance, or numerical scores, are available for each of n persons on each of n jobs, and the problem is the quest for an assignment of persons to jobs such that the sum of the n scores so obtained is a maximum. [1], [2], [3]

The assignment problem is itself a special case of the classical "transportation" problem encountered in Operations Research, [4] which is stated as follows:

Determine $X_{ij} \geq 0$ which minimize (or maximize)

$$\sum_{i,j} c_{ij} X_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad (i = 1, 2, \cdots, m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad (j = 1, 2, \cdots, n)$$

$$\sum_{i} a_i = \sum_{j} b_j$$

where a_i, b_j, and c_{ij} are given constants.

The transportation problem reduces to the assignment problem when

$$a_i = b_j = 1, \text{ and } m = n$$

The assignment model may be applied to the scheduling problem in the following manner:

let

$$x_{ij} = \begin{cases} 1, & \text{where unit i is scheduled at time j} \\ 0, & \text{where unit i is not scheduled at time j} \end{cases}$$

$$c_{ij} = \begin{cases} 1, & \text{where the scheduling of unit i at time j is permissible} \\ 0, & \text{where unit i may not be assigned time j} \end{cases}$$

$$\sum_i a_i = \sum_j b_j = \sum_{i,j} c_{ij} x_{ij}$$

a_i represents the number of hours that unit i is to be scheduled. There is no restriction on b_j, other than $0 \leq b_j \leq m$, where m is the number of units to be scheduled.

The schedule would be complete, not explicitly for $\sum_{i,j} c_{ij} x_{ij}$ a maximum, but when the required a_i are obtained.

The difficulty in this model lies in establishing the c_{ij}. The assignability of a unit is a function of the time periods that the unit has already been assigned, and a function of the times that another unit with an identical parameter has been assigned. If the index i is numerically the order of assigning units to the schedule,

$$c_{ij} = f\left(x_{i,j-1}, x_{i,j-2}, \ldots, x_{i,1}, x_{i-1,j}, x_{i-2,j}, \ldots, x_{1,j}\right)$$

This relationship can be delineated as a function of the given constraints; however, the order of complexity of this sub-problem compares with that of the initial problem. Therefore, the methods of linear programming have not been investigated further as a desirable approach to the scheduling problem.

2.2 Search.

If, for a given problem, there exists a means for checking a proposed solution, the problem can be solved by testing all possible answers. If there exists a solution to such a problem, that solution can be found eventually by any exhaustive process which searches through all possibilities. But the search is normally too inefficient for practical use. A search of all of the paths through the game of checkers involves some 10^{40} move choices [5]; in chess, some 10^{120} [6]. In scheduling 100 units to an array of 45 time periods, there are some 10^{600} possibilities. Clearly, the search method is not a practical method of attacking the scheduling problem.

2.3 Heuristics

Since the goal is to determine only one of many possible optimum solutions, any practical method tending to reduce the search space would be desirable The constraints provide some reduction in the total number of possibilities, but normally the remaining number is too great for search.

The nature of a complex scheduling problem is such that a vast number of variables exist for which no mathematical model or algorithm is practical. It is a symbol manipulating process where the human scheduler uses such terms as "likely," "doubtable," or "seem to" in steps toward solution. Moreover, some features of the schedule may be "desirable" but not essential.

It is possible to simulate the human scheduler's behavior by attempting first those possibilities that appear "likely" to lead to more promising results while postponing consideration of those that do not "seem" promising. These loosely defined qualifications are the "hints" that provide the basis for a heuristic approach to the problem. Proper application of heuristics can rule out whole classes of the search space.

3. Heuristics in Problem-solving

Heuristics is the term applied to a problem-solving procedure that utilizes a collection of information and experience about the nature of a problem to:

a. suggest the order in which possible solutions should be examined, and

b. direct the pattern of search in promising directions.[1]

The employment of heuristics to solve the scheduling problem is an attempt to apply the same procedures in solving the problem with a

[1] Minsky /7/ defines the adjective heuristic as "related to improving problem-solving performance." The noun is used "in regard to any method or trick used to improve the efficiency of a problem-solving system." Newell, Simon, and Shaw /8/ state that heuristics are "things that aid discovery. Heuristics seldom provide infallible guidance; they give practical knowledge possessing only emperical validity. Often they 'work', but the results are variable and success is seldom guaranteed."

computer that a human would apply. It is not intended that the heuristics program be general enough for application beyond this one problem. Moreover, the connection between heuristics and learning is not made in this application; i.e., the program is not designed to be self-improving as experience is gained.[2]

3.1 Means-ends Analysis of General Problem Solver.

The principle of the General Problem Solver (GPS) of Newell, Simon, and Shaw [5] for modeling human mental processes represents a practical approach to the application of heuristics to machine problem-solving. The means-ends analysis of the GPS is a procedure designed for problems that can be expressed in terms of objects and operators. An operator is something that can be applied to certain objects to produce different objects. The objects can be characterized by the features they possess, and by the differences that can be observed between pairs of objects.

As the name implies, the GPS was designed as a general program, independent of problem type. It has been successfully employed to solve problems in symbolic logic, proof of some trigonometric identities, [8] and chess playing [9]. The specifics of the problem under consideration, including definitions of objects, operators, the rules for applying and the order of applying operators, and the like, constitute the task environment. The task environment defines the rules of the

[2] Significant work is proceeding to implement basic learning heuristics supported by success-reinforced decision models in "learning" programs. [5], [6], [9]

particular problem, the GPS program defines the procedure for applying the information in the task environment.

One method of the means-ends analysis of a problem that is applicable to the scheduling problem is shown in Fig. 1, which characterizes the GPS goal-type #1 as the prime goal, with goal-type #3 as a subgoal. The prime goal is to transform object a into object b. If possible in the present state, success is immediately achieved; otherwise, the difference d between the objects is detected and the type #3 subgoal is called uponto findan applicable operator for reducing d. If no operator exists to reduce d, failure occurs; otherwise the operator q is applied to d and the subgoal is achieved, resulting in the new object c. Now the prime goal is to transform the new object c into b, and the procedure is repeated with c replacing a.

The term difference need not bear the mathematical connotation of subtraction. Difference between objects is some measure of the relationship of an attribute of the current object to the attribute of the desired object. Difference is defined in the task environment and is particular to the problem.

3.2 Means-ends Analysis of the Scheduling Problem.

The means-ends analysis of the GPS can be applied at two levels in the scheduling problem.

At the first or lower level, corresponding to the first functional block (a) of Fig. 1, this analysis is used to assign units to the more desirable elements of the schedule. Analysis at this level is shown

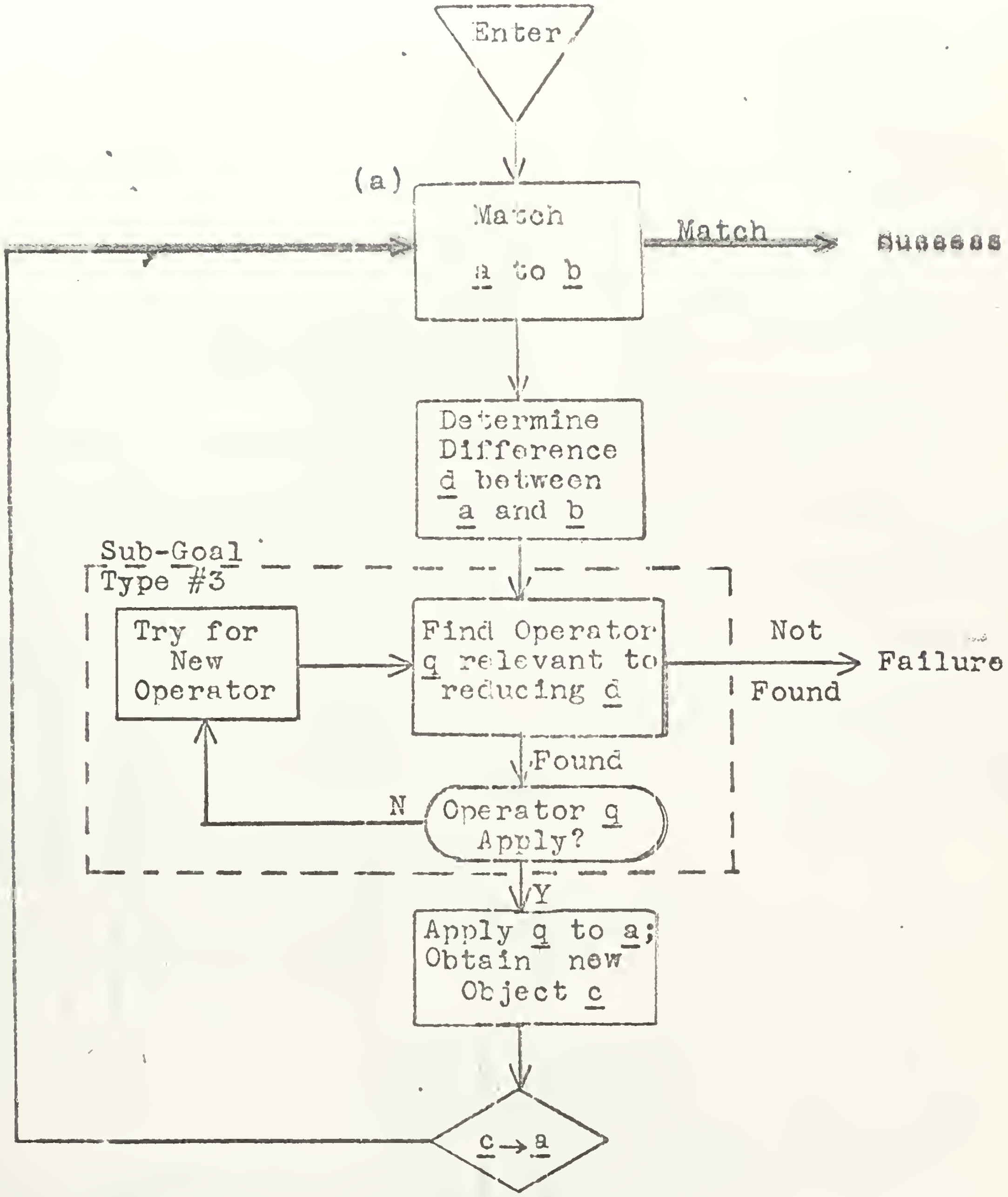


Figure 1

General Problem Solver goal type #1; Transform Object a to Object b.

in Fig. 2. Since, in general, all units cannot be assigned the desirable times or the problem would be trivial, a hierarchy of desirability and order of assigning units are required as part of the task environment. The units attempted first will receive the most desirable times consistent with the constraints; units attempted last will, in general, receive less desirable assignments in the schedule.

As the assignment of units proceeds in the non-trivial problem, a unit will be found that cannot be scheduled in a desirable pattern because of conflicting parameter with a unit already assigned. Another pattern of assignment must be sought which, although at less desirable times, is still consistent with the constraints. This procedure continues until there are no more sets of available times because of conflict or constraints. This condition is termed total conflict.

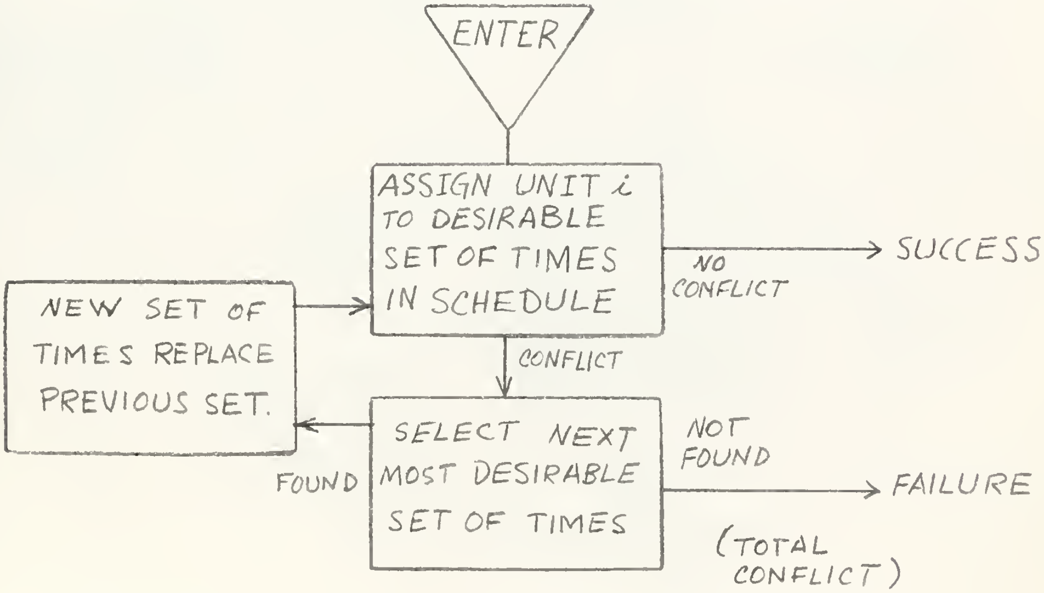

ENTER
ASSIGN UNIT i TO DESIRABLE SET OF TIMES IN SCHEDULE
NO CONFLICT
SUCCESS
NEW SET OF TIMES REPLACE PREVIOUS SET.
CONFLICT
SELECT NEXT MOST DESIRABLE SET OF TIMES
FOUND
NOT FOUND
FAILURE
(TOTAL CONFLICT)

Figure 2.
Goal: Assign unit i to schedule without modification of previous assignments

With reference to Fig. 2, the most desirable times for unit i correspond to object a of Fig 1, and object b is a set of available times in the schedule for the unit being assigned. The operator applied to a when assignment is prohibited is the process of selecting another set of times in the hierarchy of desirable times that is consistent with the constraints on the unit. The result is either failure or a replacement of a with the new set of times, c.

The second or higher level analysis, shown in Fig. 3, is applied on failure of the first level (total conflict). In order for unit i to be successfully assigned, a set of previously assigned units must be selected for deletion from the schedule and reassigned.

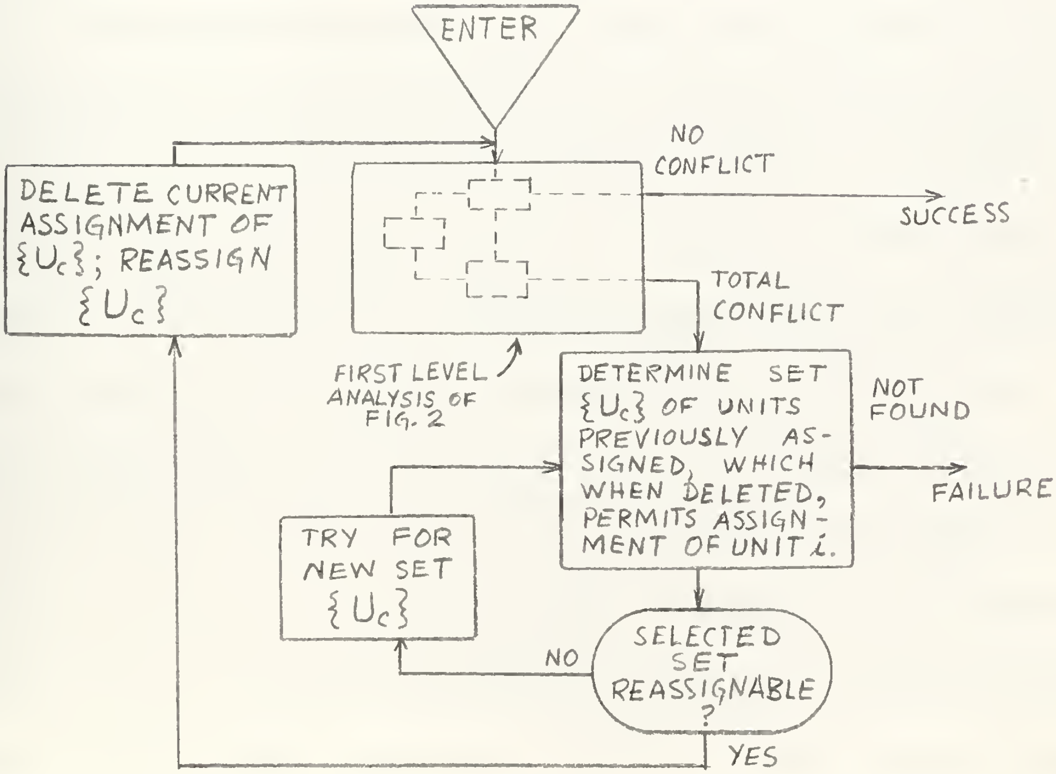

ENTER
NO CONFLICT
SUCCESS
DELETE CURRENT ASSIGNMENT OF {U_c}; REASSIGN {U_c}
TOTAL CONFLICT
FIRST LEVEL ANALYSIS OF FIG. 2
DETERMINE SET {U_c} OF UNITS PREVIOUSLY ASSIGNED, WHICH WHEN DELETED, PERMITS ASSIGNMENT OF UNIT i.
NOT FOUND
FAILURE
TRY FOR NEW SET {U_c}
NO
SELECTED SET REASSIGNABLE ?
YES

Figure 3.
Goal: Assign unit i to schedule;
Reassign other units if necessary

Failure at the second level as it now stands, occurs once it is found that there are no more reassignable sets $\{U_c\}$ that would permit assignment of the current unit, U_i.

The objects and operators have different meanings in the second level than in the first. The first level now takes the functional position of "match". Object a is an allowable set of times in the schedule, and object b is the current unit to be assigned. The operator sought upon failure is again a process of selection, in this case the selection of units to be reassigned. The result is a modified set, c, of available times to replace a.

3.3 Backtracking and the Heuristic Measure of Complexity.

The second level analysis may now be expanded upon failure by employing the method of backtracking as described by Golomb /10/. Let $\{U_c\}_1$ be the initial set considered for deletion. U_i may be replaced in Fig. 3 by $\{U_c\}_1$ and a set $\{U_c\}_2$ will be sought for deletion such that $\{U_c\}_1$ may be reassigned, and U_i again attempted for assignment. If this fails, another set $\{U_c\}_2$ will be sought, and so on until there exist no more sets $\{U_c\}_2$ available for deletion. In this event, the second set of $\{U_c\}_1$ initially tried replaces the first set of $\{U_c\}_1$, and the procedure is repeated. In the event all sets $\{U_c\}_1$ are attempted without determination of a reassignable set $\{U_c\}_2$, then the first set of $\{U_c\}_2$ selected replaces U_i and a third level backtrack is attempted to seek a reassignable set $\{U_c\}_3$. The backtrack procedure is shown in Fig. 4, where the subscript ℓ

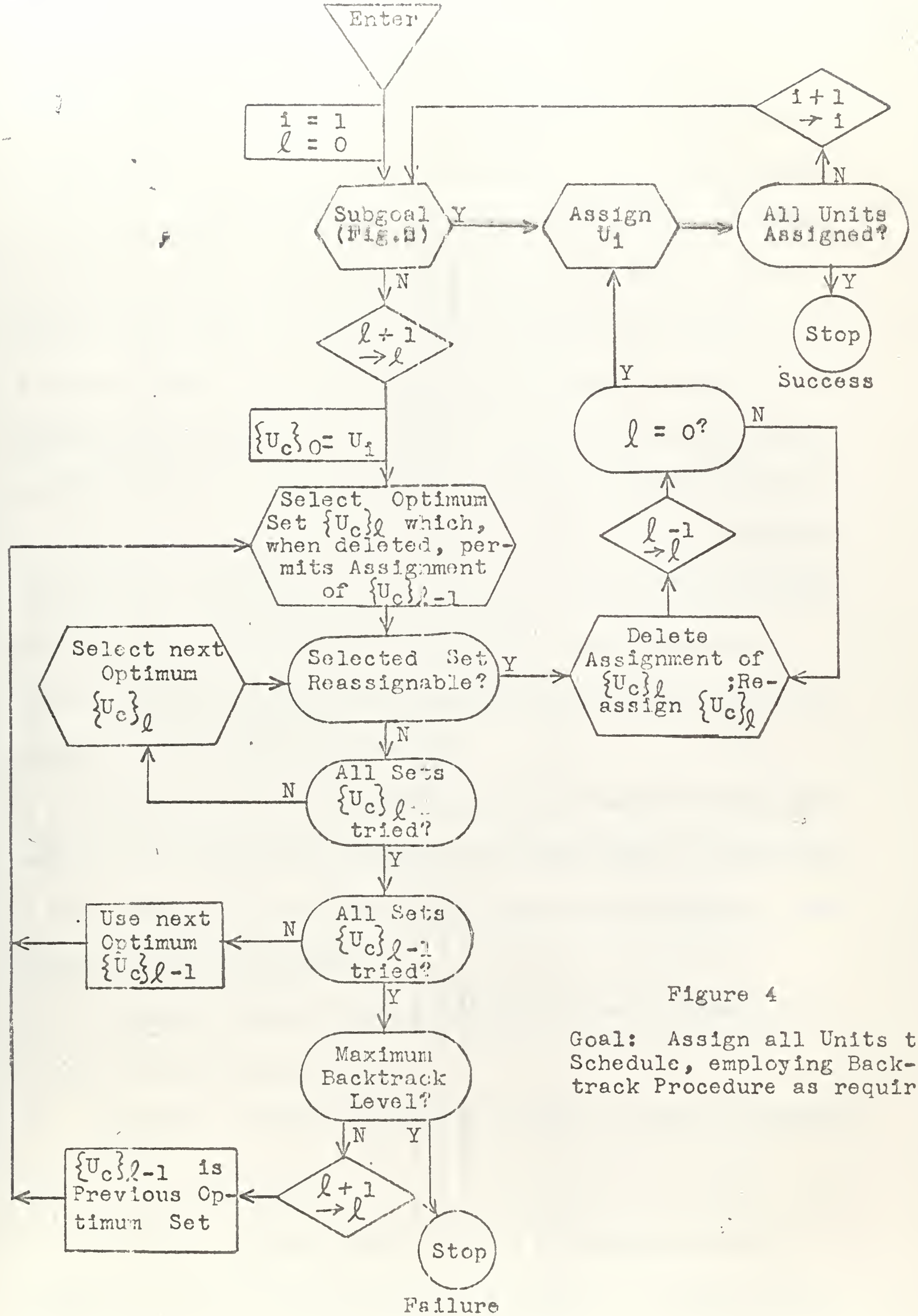

Enter
i = 1
ℓ = 0
Subgoal (Fig.8)
Y
Assign U_i
All Units Assigned?
N
i + 1 → i
Y
Stop
Success
N
ℓ + 1 → ℓ
{U_c}_0 = U_i
Select Optimum Set {U_c}_ℓ which, when deleted, permits Assignment of {U_c}_{ℓ-1}
Y
ℓ = 0?
N
ℓ - 1 → ℓ
Select next Optimum {U_c}_ℓ
Selected Set Reassignable?
Y
Delete Assignment of {U_c}_ℓ ; Reassign {U_c}_ℓ
N
All Sets {U_c}_ℓ tried?
N
Y
Use next Optimum {U_c}_{ℓ-1}
All Sets {U_c}_{ℓ-1} tried?
N
Y
Maximum Backtrack Level?
N
Y
{U_c}_{ℓ-1} is Previous Optimum Set
ℓ + 1 → ℓ
Stop
Failure

Figure 4

Goal: Assign all Units to Schedule, employing Backtrack Procedure as required.

represents the backtrack level, and a maximum level is given.

The success of the first attempt at assignment or subsequent reassignment of a unit will depend to a great extent on the initial order in which the units are considered. Those units with the largest number of potentially conflicting parameters, with the most constraints, and required to be assigned to the most number of times in the schedule, are the most complex. Let a heuristic measure of complexity based on these factors be assigned each unit. The most complex unit, by this measure, will be less flexible in the schedule and harder to assign or reassign as the array fills up, while the least complex unit is more easily assigned to a crowded schedule than the most complex. The initial assignment of units will be in order of decreasing complexity, and the selection of units for deletion and reassignment in the backtrack operation will be a function of this measure of complexity.

The efficiency of the backtrack operation is based on the fact that once it is found that a sub-assembly of the search space is incompatible, a large number of conceivable assemblies need never be examined, thereby reducing the search space.

Consider a specific scheduling problem; scheduling classes at the U. S. Naval Postgraduate School.

4. Case Study, Scheduling of Classes at the U. S. Naval Postgraduate School.

4.1 The Task Environment.

The class schedule at the U. S. Naval Postgraduate School is designed to satisfy all given academic requirements. The requirements for all students are known prior to constructing the schedule, and the schedule is designed to reflect these requirements. [1,2] An optimum schedule is one that is non-conflicting and that satisfies academic requirements and constraints.

The scheduling parameters are: courses, students, instructors, and classrooms. One particular combination of these parameters (with or without classroom specification) constitutes the unit of assignment. An element of the student parameter is called a section, which is one unique set of students who take the same courses together, under the same instructors, and at the same times. The individual instructor is the element of the instructor parameter; one course and one room designation, are the elements of the other parameters.

The first three parameters of the unit must be specified; the lecture room or laboratory room is specified in the unit only if they are essential to the course. Otherwise, the computer program assigns appropriate classrooms for laboratory or lecture periods.

[1] Professor C. C. Gotlieb, Computation Centre, University of Toronto, has designed a computer program for a similar problem; scheduling of Toronto high school classes. Correspondence with Professor Gotlieb indicates that the method of solution is as yet undocumented, but is to be presented at the International Federation of Information Processing Societies, Munich, August 27-September 1, 1962.

[2] Larger universities normally prepare a schedule of courses and classes, and the students attempt to elect their courses and class meetings to fit the existing time-table according to availability of desired courses. Automatic data processing is employed at the University of Purdue for automation of student registration. /11/

The academic time-table is one school week of 45 one-hour periods; nine periods a day, five days a week. A concurrent time table is prepared by the program to provide one half day a week (periods one through five, or five through nine) for students that are required to maintain flying proficiency in addition to academic requirements. The maximum number of students that may fly in each half-day (or each fly period) is dependent upon the availability of aircraft, and established by the U. S. Naval Air Facility, Monterey. This quota may vary between days of the week, and between school quarters.

4.2 Data Design.

The following data constitutes the input to the program.

(1) SECTLIST: a table of four fields, each entry of the table consists of the following items:

(a) section designation

(b) number of students in the section

(c) number of students that are to be assigned a fly period

(d) courses to be taken by the section

(2) CORSHORS: a table of four fields, with each entry consisting of the following items:

(a) course designation

(b) number of lecture periods required for the course

(c) number of laboratory meetings required for the course

(d) number of hours for each laboratory meeting.

(3) ROOMLIST, a table of three fields, each entry consisting of the following items:

(a) room designation

(b) number of students that can occupy the room

(c) number of units that may occupy the room concurrently, especially in the case of large laboratory spaces if facilities are sufficient.

(4) FLYQUOTA: a two by five array corresponding to each fly period; each element containing the maximum number of students to be assigned to the fly-period.

(5) CNSTRNTS: a variable-field table with a variable number of items per field. This table contains in the leading field, a parameter for which specific constraints exist, and in subsequent fields, the times or other parameters constrained.

The constraints are divided into two types. Those that require a pairing of parameters, or pairing of parameters with time, are called positive constraints. Those constraints requiring that parameters, or parameters and time, not be paired are called negative constraints. Each type is subdivided further into priority one and priority two constraints. Priority one constraints are those for which a firm requirement exists and must be satisfied. Priority two constraints are to be satisfied if possible; however, these constraints will be removed if necessary to prevent total conflict.

(6) UNITLIST: the units of assignment, a table of four fields, the first containing one item, and the others are of variable length. The last field may be void.

(a) course designation (α)

(b) section(s) designations (β)

(c) instructor(s) designations (γ)

(d) room(s) designations (δ)

A UNITLIST entry, or one unit, will be referred to frequently in the remainder of the report. A unit will be represented by U_i and will be considered to mean its corresponding entry in UNITLIST, represented as follows.

$$U_i = \{ \alpha_i ; \beta_j \cdots \beta_k ; \gamma_l , \gamma_m^a , \gamma_n^b ; \delta_p^a , \delta_q^b \}$$

The superscripts a or b indicate lectures only or laboratories only respectively, for items γ and δ (instructors and classrooms respectively). An instructor (γ) without superscript implies a requirement for scheduling to all laboratories and lectures of the unit. If a classroom (δ) is specified in the unit, it must carry an indication for lecture or laboratory, and therefore must be represented with a superscript.

In order to continue previous procedures employed at the school in schedule preparation, there is a Phase I to the program for initial processing of the input data SECTLIST and CORSHORS. Its output is a listing of sections by course to assist the academic staff in grouping sections into units and in assigning instructors to the units. Phase I, therefore, is processing to assist preparation of the UNITLIST data.

This phase does not involve complex processing procedures and is not described in this report

4.3 General Constraints and Optimum Backtrack Sets.

The specific constraints on a given quarter's schedule are those in the CNSTRNTS data. General constraints are independent of the quarter being scheduled and are the rules governing the first level analysis. They are the guidelines that establish the hierarchy of desirability, leading to what might be called a "good" schedule. The general constraints are listed below.

a. Lecture and laboratory periods are not to be scheduled on the same day, unless the total number of periods for the unit exceeds five; except to prevent total conflict.

b. Laboratory periods are to be scheduled in the afternoon where possible, and lecture periods in the mornings.

c. All sections are to be free during fourth or fifth periods for lunch. Professors should have one of these periods free if possible.

d. Where possible, one period following laboratory sessions is to remain unassigned in order to complete experiments that may run overtime.

e. Flyers assigned morning fly periods are not available for further assignment that day until sixth period; flyers assigned afternoon fly times are not available otherwise after fourth period.

f. A course that meets a relatively few times a week is to have assignments "spread out" over the week.

g. As described for CNSTRNTS data, priority two specific constraints are to be considered and satisfied if feasible. Otherwise, they will be deleted to prevent total conflict. Priority two specific constraints correspond to the desirable features in the general constraints.

h. Where possible, each unit should be assigned to the same period and classroom for all lecture meetings of the unit in the week.

i. A section should be assigned to the same lecture room for as many of its courses as practicable to maintain a low level of between class changes. It is undesirable for sections to change buildings between classes except when essential.

j. Lecture periods allowable for assignment are periods one through six; laboratory periods may be scheduled any required number of consecutive periods in the day. However, late afternoon laboratory periods are undesirable.

It is apparent that all the above features cannot be effected in a schedule. However, all are to be considered and as many incorporated in the schedule as determined feasible without unnecessary extension of the search space. Most of the features will be reflected in assignment of earlier units. No general constraint will be a reason for total conflict; all (except (e) which is firm) will be removed prior to backtracking. By definition, a schedule with a few general constraints satisfied is still an optimum solution, though not as "good" as one that satisfies many.

The second level analysis called in the event of total conflict requires a heuristic for selecting a set of previously assigned units with good chance of successful reassignment when deleted. Let the units be ordered by the index i in decreasing complexity as defined in section 3.3. A selection of a set of units for deletion will be a function of i and the number n of units in the set. The larger the index i and the smaller the number n, the easier the reassignment can be effected. Therefore, a modified minimax strategy suffices for this heuristic and is applied as follows:

(a). determine all periods, k, for which a conflict exists, and the units that cause the conflict,

(b). let y_k be the minimum unit number (the unit with the minimum index i) of the conflicting units in each period k,

(c). let n_k be the number of conflicting units in each period,

(d). select k such that

$$x_k = y_k/n_k \text{ is a maximum}$$

The larger the x_k, the greater the probability of reassignment of the set of units assigned period k. The set of units $\{U_c\}$ and the periods k for which x_k is a maximum will be considered the optimum backtrack set and period for deletion and reassignment in order to assign the current unit for which total conflict has been determined.

4.4 Source-Language for program.

A heuristic process is basically a non-mathematical process although numerical representation of data or an attribute may be used.

In general, heuristics employ symbols for these representations, and the process is one of symbol manipulation.

Ideally, a symbol manipulating program for digital computer is best programmed for a list processor compiler such as Information Processing Language V (IPL-V) [13] or LISP [14]. List processors have the property of maintaining links between items in a list rather than the more common method of using sequential storage locations of the specific machine. A list processor permits a flexible means of listing symbols with respect to certain attributes or characteristics as well as a variable order of the list through modification of the list linkages. [12] A language designed for numerical computations is not easily utilized for symbol manipulation without a complex or inefficient executive program.

A list processor is not available for the U. S. Naval Postgraduate School's Computer Center since the Center is primarily employed for numerical scientific computation. One available compiler, the CDC 1604 FORTRAN [15], has provision for Boolean expressions which may be effectively used for operations on symbols and describing their attributes.

A set of Boolean arrays, each representing a dichotomous attribute, can be used to describe a series of attributes of any symbol. Boolean operations on the input data provide a means of constructing the attribute arrays. Consider the following example Let the Boolean matrix (C) with elements c_{ik} represent the presence ($c_{ik} = 1$) or absence ($c_{ik} = 0$) of a constraint on unit i at period k. Similarly, let u_{ij} be an element of the conflict matrix (U) with the value 1 or 0 depending upon whether or not there exists a common parameter in unit i and unit j. Let t_{jk}, an element of (T), indicate whether or not unit j has been assigned period k.[1] Then, if it is desired to determine whether unit i may be assigned to period k as a function of constraints (C), and conflicts (U), determine

$$w_{ik} \overset{\mathcal{L}}{=} c_{ik} + \sum_{j} (u_{ij}) \cdot (t_{jk})$$

where $\mathcal{L}$ indicates a Boolean or logical expression meaning that summations are logical "or's" and products are logical "and's". Therefore,

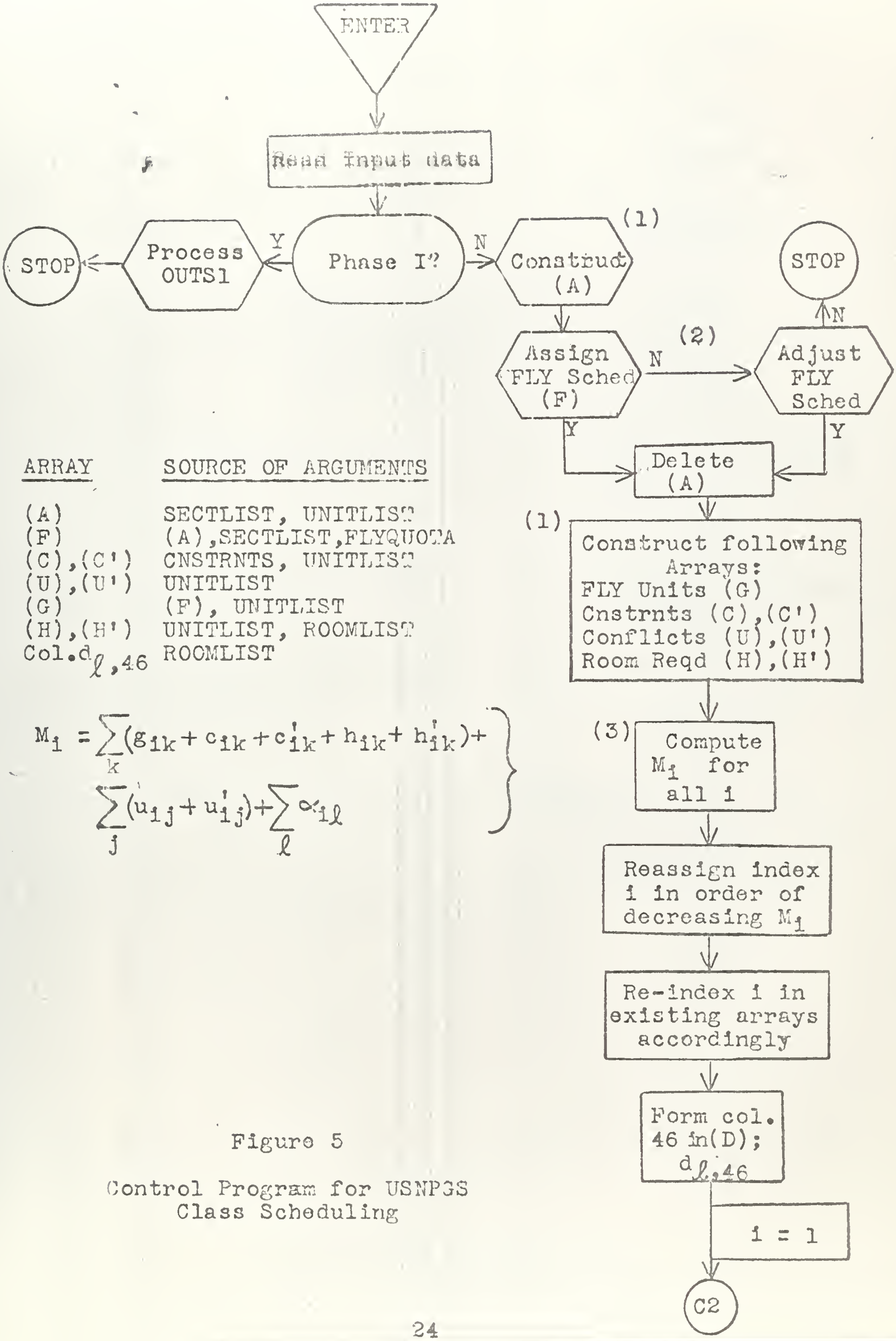
$$w_{ik} = \begin{cases} 1; & \text{for unit i having conflicting parameter with another unit already assigned period k, or unit i is constrained at period k} \\ 0; & \text{otherwise} \end{cases}$$

The computer program and its description that follow have been designed to be largely independent of machine and language. However, terminology used does reflect Boolean descriptions for ease in programming. It is not limited to languages that employ Boolean operations as long as a means exists for determining equivalent attributes as those that are described.

4.5 Computer program.

Fig. 5 is a flow chart representation of the control program for the scheduling of classes at the U. S. Naval Postgraduate School. The

[1] In this case, (T) is the current status of the schedule.



ARRAY	SOURCE OF ARGUMENTS
(A)	SECTLIST, UNITLIST
(F)	(A),SECTLIST,FLYQUOTA
(C),(C')	CNSTRNTS, UNITLIST
(U),(U')	UNITLIST
(G)	(F), UNITLIST
(H),(H')	UNITLIST, ROOMLIST
Col.$d_{\ell,46}$	ROOMLIST

$$M_i = \left.\sum_k (g_{ik} + c_{ik} + c'_{ik} + h_{ik} + h'_{ik}) + \sum_j (u_{ij} + u'_{ij}) + \sum_\ell \alpha_{i\ell} \right\}$$

Figure 5

Control Program for USNPGS Class Scheduling

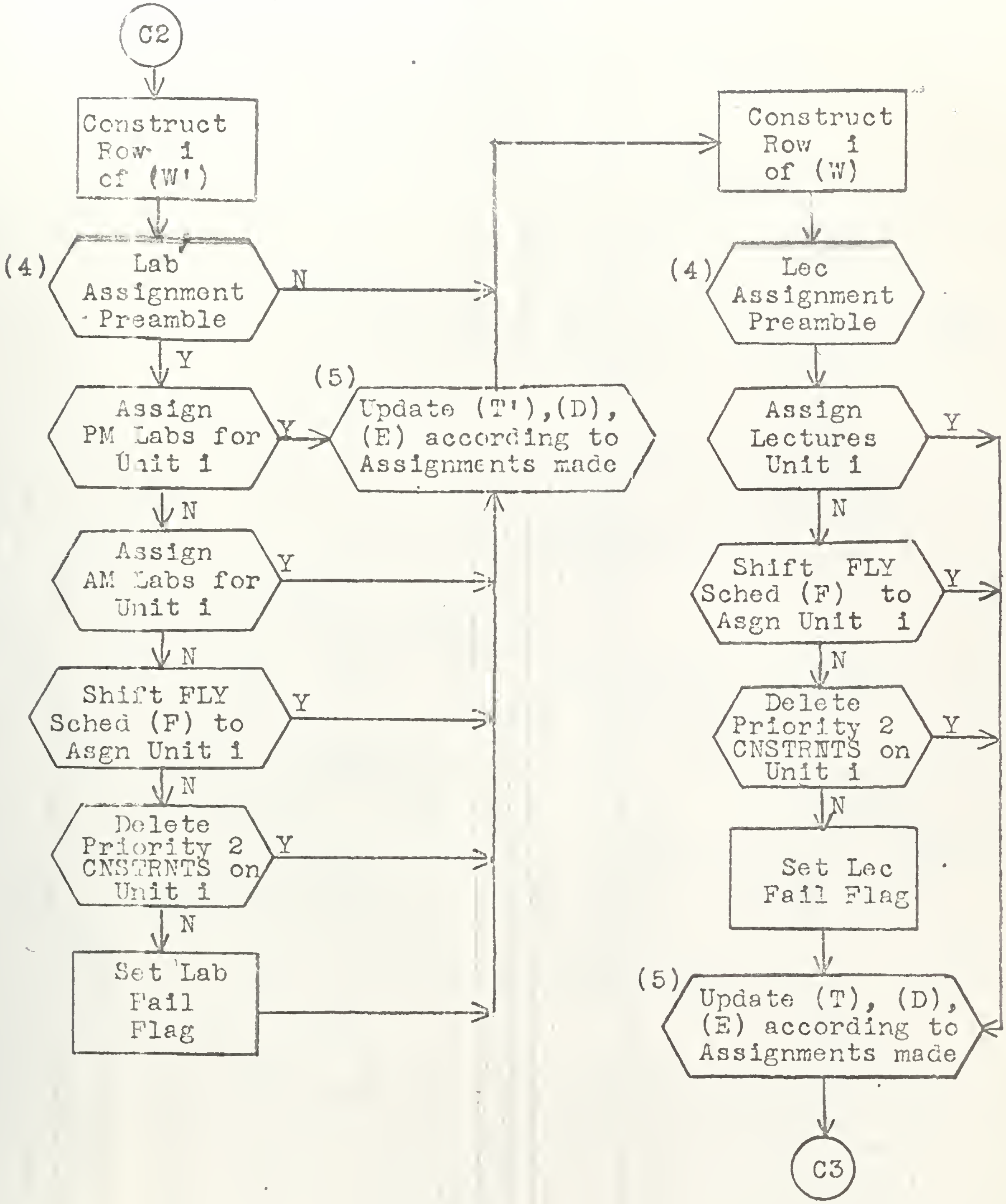

C2
Construct Row i of (W')
(4)
Lab Assignment Preamble
N
Y
Assign PM Labs for Unit i
Y
(5)
Update (T'),(D), (E) according to Assignments made
N
Assign AM Labs for Unit i
Y
N
Shift FLY Sched (F) to Asgn Unit i
Y
N
Delete Priority 2 CNSTRNTS on Unit i
Y
N
Set Lab Fail Flag
Construct Row i of (W)
(4)
Lec Assignment Preamble
Assign Lectures Unit i
Y
N
Shift FLY Sched (F) to Asgn Unit i
Y
N
Delete Priority 2 CNSTRNTS on Unit i
Y
N
Set Lec Fail Flag
(5)
Update (T), (D), (E) according to Assignments made
C3

Figure 5
(cont'd)

Control Program for USNPGS
Class Scheduling

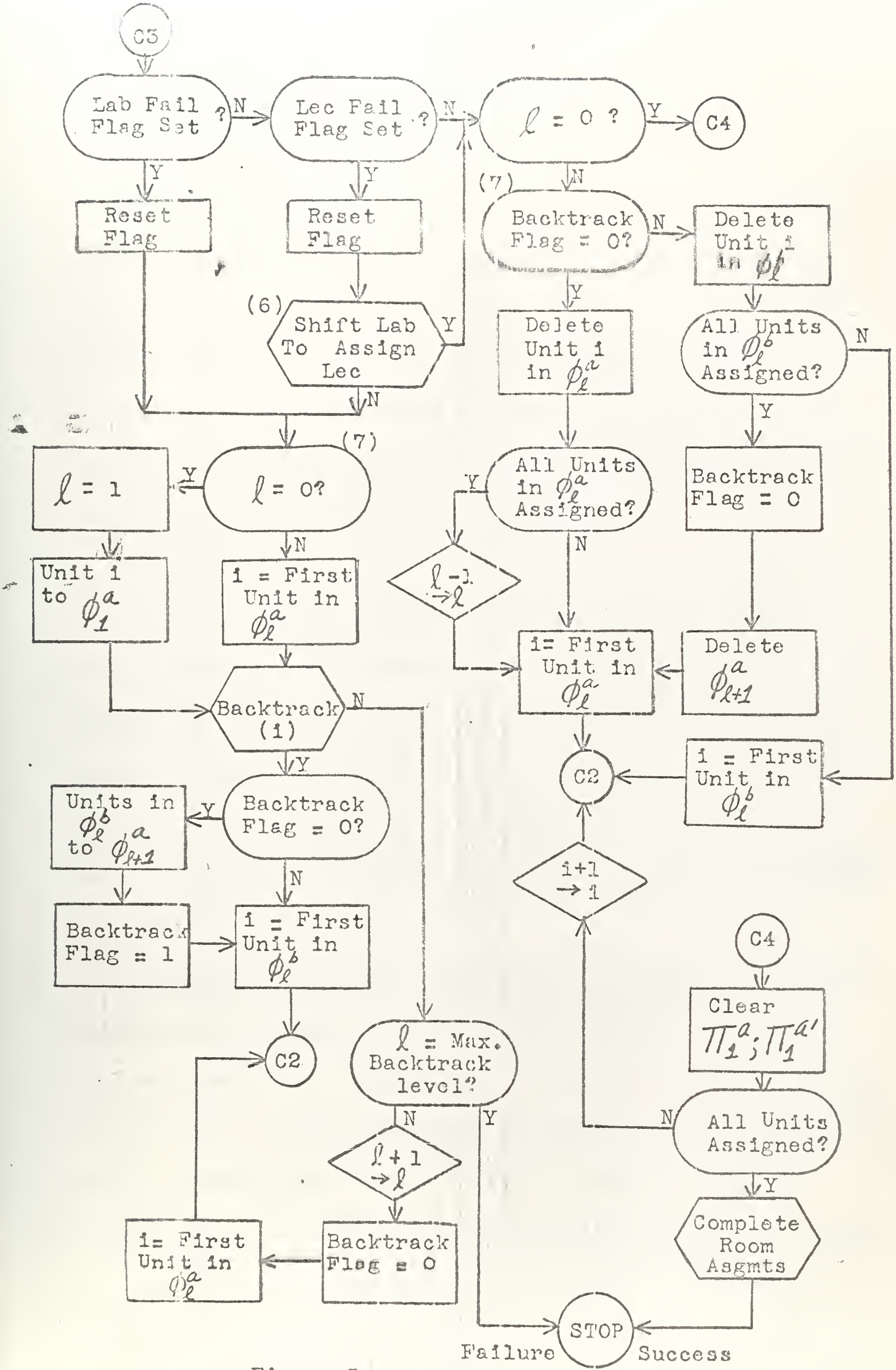

C3
Lab Fail Flag Set ?
N
Lec Fail Flag Set ?
N
ℓ = 0 ?
Y
C4
Y
Reset Flag
Y
Reset Flag
(7)
N
Backtrack Flag = 0?
N
Delete Unit i in ϕ_ℓ^b
(6)
Shift Lab To Assign Lec
Y
Y
Delete Unit i in ϕ_ℓ^a
All Units in ϕ_ℓ^b Assigned?
N
N
Y
(7)
ℓ = 1
Y
ℓ = 0?
Y
All Units in ϕ_ℓ^a Assigned?
Backtrack Flag = 0
N
N
$\ell - 1 \rightarrow \ell$
Unit 1 to ϕ_1^a
i = First Unit in ϕ_ℓ^a
i= First Unit in ϕ_ℓ^a
Delete $\phi_{\ell+1}^a$
Backtrack (i)
N
Y
C2
i = First Unit in ϕ_ℓ^b
Units in ϕ_ℓ^b to $\phi_{\ell+1}^a$
Y
Backtrack Flag = 0?
N
i+1 → i
Backtrack Flag = 1
i = First Unit in ϕ_ℓ^b
C4
Clear $\Pi_1^a ; \Pi_1^{a'}$
C2
ℓ = Max. Backtrack level?
N
Y
N
All Units Assigned?
$\ell + 1 \rightarrow \ell$
Y
Complete Room Asgmts
i= First Unit in ϕ_ℓ^a
Backtrack Flag = 0
STOP
Failure
Success

Figure 5
(cont'd)

control program indicates the order of calling the subprograms contingent upon the status of the conditions sensed in decision blocks. Each subprogram represented by a hexagon block in Fig. 5 is a major subprogram, either a function or a subroutine, shown in detail in appendix A. Some of the subprograms terminate with a decision and provide two exits. Functions of lesser complexity are indicated in rectangular blocks. The descriptions below pertain to the control program, where the corresponding number of the description is shown adjacent to the appropriate block in Fig. (5).

(1) Arrays.

Arrays are used to delineate attributes, as described in the previous section, and are represented by a capital letter in parenthesis; e.g., (C) or (C'). A primed letter indicates an array with the same attributes for laboratory assignments as the unprimed mate represents for lecture assignments. An unprimed letter without a primed mate is used if there is no distinction between lectures and laboratories in the attribute.

An array that describes an attribute as determined by direct processing of the input data without reflecting a decision by the program will be referred to as a constant array. A constant array is not modified in the program. An array that describes attributes that are derived as a result of program's decision and modified if necessary, is called a variable array. Not all arrays require construction although the attribute they represent must be available; they are described for convenience to simplify the logical expressions for which the attributes are treated as independent variables. The arrays, the attributes they represent, and sources of data used as arguments for construction of the arrays are described below.

a. (T), (T'), These are variable arrays representing the schedule by indicating the units that have been assigned to laboratory and lecture periods. Elements of these arrays are:

$$t_{ik} = \begin{cases} 1, \text{ unit } U_i \text{ is assigned period } k, \text{ lecture} \\ 0, \text{ unit } U_i \text{ not assigned period, } k, \text{ lecture} \end{cases}$$

$$t'_{ik} = \begin{cases} 1, \text{ unit } U_i \text{ is assigned period } k, \text{ laboratory} \\ 0, \text{ unit } U_i \text{ not assigned period } k, \text{ laboratory} \end{cases}$$

b. (A), A temporary array constructed to record the frequency of occurance of all pairs of sections in different units. When two or more sections requiring assignment to the fly schedule appear in more than one unit together, conflicts that appear in the scheduling of these units are held to a minimum by assigning their common sections to the same fly period (A) is a constant array determined by direct processing of the third field of the SECTLIST table and the UNITLIST table. The element a_{ij} is the number of times sections i and j are contained as a pair in different units.

c. (F); A variable array representing the fly schedule, with element

$$f_{jm} = \begin{cases} 1, \text{ Section } \beta_j \text{ assigned fly period } m \\ 0, \text{ otherwise} \end{cases}$$

The sources of data for (F) are (A), SECTLIST, and FLYQUOTA. The fly schedule is initially arbitrary within the quotas except for assignment to the same fly periods of sections that are paired in accordance with the purpose of (A). (F) is modified later in the program to prevent total conflict, if applicable. The routine for constructing (F) is shown in Appendix A.

d. (G); the variable array of units vs academic periods indicating the existence of a section in the unit that has been assigned to (F) at the same time. An element of (G) is

$$g_{ik} \overset{\mathcal{L}}{=} \sum_{j} \beta_{ij} \cdot f_{jm}$$

where β_{ij} is section β_j in unit U_i, f_{jm} an element of (F), and k is an academic period. The source of data for (G) is UNITLIST and (F). The 10 fly periods m are related to the 45 academic periods k as shown in Table 1

Day	m	k
Monday AM	1	1-5
Monday PM	2	5-9
Tuesday AM	3	10-14
Tuesday PM	4	14-18
Wednesday AM	5	19-23
Wednesday PM	6	23-27
Thursday AM	7	28-32
Thursday PM	8	32-36
Friday AM	9	37-41
Friday PM	10	41-45

Table 1.
Relationship between days of the week, fly periods m, and academic periods k.

e. (C), (C'); the constant constraint arrays, representing the existence of specific constraints on parameters with respect to lectures or laboratories or both. If $\psi^c_{\ell k}$ represents a parameter (either $\alpha_\ell, \beta_\ell, \gamma_\ell, \gamma^a_\ell$, or $\delta_\ell{}^a$) negatively constrained at period k, and $\psi_{\ell i}$ represents that same parameter in unit U_i, then an element of (C) is:

$$c_{ik} \overset{\mathcal{L}}{=} \sum_\ell \psi^c_{\ell k} \cdot \psi_{\ell i}$$

If the constraint is positive, $\psi^c_{\ell k}$(pos), the periods k are complemented and

$$\psi^c_{\ell k} = \overline{\psi^c_{\ell k}(pos)}$$

Therefore,

$$c_{ik} = \begin{cases} 1, & \text{unit } U_i \text{ constrained at period k} \\ 0, & \text{otherwise} \end{cases} .$$

(C') is determined by the same rules as those governing the construction of (C), for laboratory constraints in lieu of lecture constraints. The sources of data for (C) and (C') are the CNSTRNTS and UNITLIST tables.

f. (U), (U'); constant conflict arrays representing presence or absence of the same sections or instructors in different units. Elements of (U) are

$$u_{ij} \overset{\mathcal{L}}{=} \sum_m \beta_{im} \cdot \beta_{jm} + \sum_\ell (\gamma_{i\ell} \cdot \gamma_{j\ell} + \gamma_{i\ell} \cdot \gamma^a_{j\ell} + \gamma^a_{i\ell} \cdot \gamma^a_{j\ell})$$

where β_{im} is section β_m in unit U_i, $\gamma_{i\ell}$ is instructor γ_ℓ in unit U_i, the superscript a indicates for assignment to lectures only. The element of (U') is identical except superscript b replaces a in the above equation. The value of u_{ij} is one if units i and j may not be assigned to the same periods, both with respect to conflicts only. The source of data for these arrays is UNITLIST.

g. (H),(H'); the room-requirement arrays that are constant and determined by direct processing of the fourth field of UNITLIST data. An element of (H) is $h_{\ell i}$, with value 1 if room δ_ℓ^a is specified for lectures of unit U_i, zero if not specified An element of (H') is $h'_{\ell i}$ with value 1 if room δ_ℓ^b is specified for laboratories of unit U_i, zero if not specified.

h. (D); a variable array representing room assignments. Element $d_{\ell k}$ is 1 if room δ_ℓ is assigned period k, and zero if not assigned period k. Column 46 of (D) (or $d_{\ell,46}$) is constant and set to one if the room δ_ℓ may be duplicated for assignment of more than one unit at one time. This is determined from the ROOMLIST table, and used in conjunction with the next array (E).

i. (E); a variable array recording the duplication of rooms when duplication is permitted ($d_{\ell,46} = 1$). $e_{\ell k}$ is one if δ_ℓ assigned twice at period k.

j. (W), (W'); variable arrays used to represent the availability of periods for assignment of a unit. These arrays utilize the information of the previously described attributes as arguments. The elements are:

$$w_{ik} \overset{\mathcal{L}}{=} g_{ik} + c_{ik} + \sum_j u_{ij} \cdot t_{jk} + \sum_\ell h_{\ell i} \cdot d_{\ell k} \cdot (\bar{d}_{\ell,46} + d_{\ell,46} \cdot e_{\ell k})$$

$$w'_{ik} \overset{\mathcal{L}}{=} g_{ik} + c'_{ik} + \sum_j u'_{ij} \cdot t'_{jk} + \sum_\ell h'_{\ell i} \cdot d_{\ell k} \cdot (\bar{d}_{\ell,46} + d_{\ell,46} \cdot e_{\ell k})$$

w_{ik} (or w'_{ik}) takes the value one when U_i may not be assigned to period k for lecture (or laboratory) because of any one of the prohibiting attributes represented by a term in the above equation.

(2). In the event that all sections requiring a fly period are not scheduled in (F) according to the desirable function of (A), and an unassigned section or sections cannot be assigned to (F) because the number of students in the section would exceed the quota of each fly period, the routine "adjust fly schedule" is called. This routine shifts some of the assignments in (F) in order that the unassigned sections may be assigned. Failure occurs at this point in the program only if the sum of the quotas for all fly periods is less than the total number of students requiring assignment to (F).

(3). M_i is a heuristic measure of complexity of a unit U_i. It is the sum of all elements generated by the input data (elements of constant arrays) known to contribute in prohibiting an assignment.

$$M_i = \sum_k (g_{ik} + c_{ik} + c'_{ik} + h_{ik} + h'_{ik}) + \sum_j (u_{ij} + u'_{ij}) + \sum_\ell \alpha_{i\ell}$$

Note that there is a similarity between terms in M_i and the terms of (W) and (W') that are elements of constant arrays. The greater the M_i, the more likely that periods k will have w_{ik} or w'_{ik} equal one, and therefore the fewer the number of periods available for the assignment of U_i.

Since the initial assignment of the index i represents only the order of the units in the UNITLIST data, it is convenient either to reorder the indexes i according to decreasing M or to assign a new index to the units and maintain a file of old and new indexes. It will be assumed in the remainder of the program that the indexes i have been reassigned and that the unit with the greatest M is now U_1.

(4). The preambles to the assignments of laboratory and lecture periods pick up the number of hours for the course from CORSHORS, and set w_{ik} and w^{0}_{ik} to one for k corresponding to periods that certain units are to be deleted during backtrack procedure. This is described in (7) below. The preambles also clear flags and indicators that may have been set in previous use of the routines.

(5) When an assignment to (T) or (T^{0}) is made, the corresponding element is set to one Variable arrays (E) and (D) are made current to reflect each room assignment, and room duplication, if any, by the following operations:

$$e_{\ell k} \overset{\mathcal{L}}{=} d_{\ell k} \cdot d_{\ell,46} \cdot (h_{\ell i} \cdot t_{ik} + h'_{\ell i} \cdot t'_{ik}) + e_{\ell k}$$

$$d_{\ell k} \overset{\mathcal{L}}{=} h_{\ell i} \cdot t_{ik} + h'_{\ell i} \cdot t'_{ik} + d_{\ell k}$$

where i indicates the unit assigned, and the operation shown is one of replacement rather than equality.

(6). The general constraint requiring that lectures and laboratories to be of different days can cause a laboratory assignment to eliminate a possible lecture day. If there are more restrictions on lecture than laboratory assignments for a unit, a shift in laboratory assignment to another day could open up a day for successful lecture assignment. The converse does not apply; laboratory periods are assigned first and are not restricted by the unit's lecture assignments. Therefore, after success on laboratory assignments and failure on lecture assignments, the routine "shift laboratory to assign lecture" is called to investigate this possibility.

(7). The remainder of the routines of the control program requires further description of the procedures and file maintenance employed in backtrack operations.

Index ℓ indicates the current backtrack level. The file Φ is a symbols cellar[1] with two variable length fields, ϕ_ℓ^a and ϕ_ℓ^b. The first field, ϕ_ℓ^a, is used to file the set of units for which the current level of backtracking is being attempted. The second field, ϕ_ℓ^b, contains successive sets of units selected by the backtrack routine for reassignment in order that units ϕ_ℓ^a may be assigned. The first set in ϕ_ℓ^b is also entered as the set $\phi_{\ell+1}^a$ in the event the next level of backtracking is required; the first set ϕ_ℓ^b will be the set backtracked on at the next level.

File Π also contains two fields; it is used to record periods k, the times for which corresponding sets in Φ have been selected for deletion. Each field Π_ℓ^a and Π_ℓ^b, are further divided into two sub-fields each to indicate periods selected for investigation as lecture or laboratory periods; e.g., Π_ℓ^a or $\Pi_\ell^{a'}$. This procedure is necessary to limit the search for periods to certain days consistent with lecture or laboratory assignments of ϕ_ℓ^a completed prior to backtracking.

[1] used as last-in, first-out files according to the method described by Samelson and Bauer /16/.

The backtrack flag is set to zero while selecting the first set ϕ_{ℓ}^{b} for reassignment and during subsequent assignments of sets in ϕ_{ℓ}^{a}. The backtrack flag is one while selecting subsequent sets ϕ_{ℓ}^{b} and attempting their reassignment. These file maintenance procedures are best described in an example.

Consider the example of successful assignments of units up through U_9, but total conflict results in attempting to assign U_{10}. ℓ is advanced to one, and U_{10} is entered as ϕ_1^a. With backtrack flag equal zero, the routine "backtrack (10)" is called, and an optimum set, say a set of only one conflicting unit, U_8 at period 20, is selected for reassignment in order that period 20 be available for U_{10}. U_8 is entered in ϕ_1^b and ϕ_2^a. k= 20 is entered in π_1^b and π_2^a.[1] The backtrack flag is now set to one and with i = 8, U_8 is deleted from the schedule and control is returned to C2 in the control program for reassignment of U_8.[2]

If successful reassignment of U_8 results, i is set again to 10 and U_{10} is assigned. The assignment of U_{10} now has been assured by the criteria used in selecting U_8 for deletion at period 20. However, if U_8 was not successfully reassigned, the routine "backtrack (10)" is again

[1] The file π is maintained in the backtrack routine. Consider in the example that only a lecture hour is needed by U_{10}, the other lecture hours and all laboratory hours having been assigned.

[2] The preamble to lecture and laboratory assignments inspects backtrack flag and periods in the appropriate π file. w_{ik} (in the example, $w_{8,20}$) is set to one so that U_8 will not be assigned again at period 20.

called and another set with associated periods are entered in ϕ_1^b and π_1^b respectively. Let this set be $\{U_4, U_3\}$ at period 22. The procedure is continued until a set selected is determined reassignable or until no more sets are found that would satisfy the requirement for completing the assignment of U_{10}. The latter condition results in the "no" exit from the backtrack routine, and the first backtrack level has been completed without success.

The index ℓ is advanced to two, backtrack flag is set to zero, and U_8 is now the set ϕ_2^a to be backtracked on. The processing continues in exactly the same manner as in the first backtrack level, except that now the periods in π_2^a (k = 20) must be eliminated from consideration in the selection of a set ϕ_2^b for reassignment for U_8 to be reassigned, in order that ultimate assignment of U_{10} is assured.

The files Φ and Π with the entries used in the example are shown in Fig. 6.

	Φ		Π	
ℓ	ϕ_ℓ^a	ϕ_ℓ^b	π_ℓ^a	π_ℓ^b
1	$\{U_{10}\}$	$\{U_8\}, \{U_4, U_3\}, \cdots$	–	$\{20\}, \{22\}, \cdots$
2	$\{U_8\}$	$\{U_5, U_6\}, \cdots$	$\{20\}$	$\{30, 35\}, \cdots$
3	$\{U_5, U_6\}$	$\cdots$	$\{30, 35\}$	$\cdots$
$\vdots$	$\vdots$		$\vdots$	

Figure 6.

Example files ϕ and π.

Failure to complete the schedule occurs when the index is advanced to L, the maximum backtrack level desired.

The backtrack routine, with subroutines OPTSET and OPTSET1, selects a heuristically optimum set of conflicting units already assigned, to try as a set in ϕ_{ℓ}^{b} for reassignment, according to the reassignment heuristic described in section 4.3 for the second level analysis.

5. Conclusions.

Heuristics in data processing is a relatively new approach to problems where the primary operation are non-numeric but are manipulations of the symbols that represent variables. The scheduling problem is one that fits this category.[1]

The maximum backtrack level in this case study has not been specified. An analysis of the number of backtrack operations used in the program, over a series of runs, will permit determination of the distribution of backtrack operations and assist in establishing a practical level.

In event of failure in a run of the program, consideration should be given to employment of the incomplete schedule that was derived.

[1] At the Spring Joint Computer Conference held in San Francisco, May 1-3, 1962, the writer had an opportunity to talk with Professor Fred Tonge (who worked with Newell in writing I.P.L.-V [13] about the application of heuristics in this problem. Professor Tonge indicated that this was probably a valid approach and made reference to Dr. William Gere at Yale University who has successfully employed heuristics to the job-shop scheduling problem. At the time of this writing, correspondence with Dr. Gere has been initiated to determine his degree of success with this approach, and to request reference to documentation, if any, of his work.

This system incorporates a natural debugging facility in that any methods used to complete the schedule manually can be incorporated in the program.

The heuristic measure of complexity of a unit, M_i, is given with its arguments having equal weight. Weighted variables may prove to be more appropriate. An area of investigation that would benefit the program is the addition of a "learning" feature in which the program, over a period of runs, would reinforce the weighted variables in M_i in successful assignments.

BIBLIOGRAPHY

1. Thorndike, R. L., "The Problem of Classification of Personnel," Psychometrika 15 (1950) 215-235.

2. Votaw, D. F., and Orden, A., "The Personnel Assignment Problem," Symposium on Linear Inequalities and Programming, SCOOP 10, USAF (1952) 155-163.

3. Kuhn, H. W., "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, NAVEXOS P-1278 March-June, 1955, 83-97.

4. Ackoff, R. L., Progress in Operations Research, Vol. I., John Wiley and Sons, ORSA #5, 1961, 111-117.

5. Samuel, A. L., "Some studies in machine learning using the game of checkers," IBM Journal of Research and Development, Vol. 3, July 1959, 211-219.

6. Shannon, C. E., "Programming a digital computer for playing chess," The World of Mathematics, Newman, Ed., Simon and Schuster, Inc., Vol. 4, 1956.

7. Minsky, M., "Steps Toward Artificial Intelligence," Proc. IRE, January 1961, Vol. 49, No. 1., 8-30.

8. Newell, A., Shaw, J. C., and Simon, H. A., "A General Problem-Solving Program for a Computer," paper given at the International Conference on Information Processing, Paris, France, June 13-20, 1959, reprinted in Computers and Automation, July, 1959, 10-17.

9. Newell, A., Shaw, J. C., and Simon, H. A., "Chess-playing and the problem of complexity, "IBM Journal Research and Development, Vol. 2, October, 1958.

10. Golomb, S., "Mathematical Theory of Discrete Classification," Fourth London Symposium on Information Theory, paper presented at the Royal Institution, Lond, held Aug. 29-Sept. 2, 1960.

11. Blakesley, J. F., "Automation in College Management," College and University Business, Nov. 1959, pp 37.

12. Green, B. F., Jr., "Computer Languages for Symbol Manipulation," I.R.E. Transaction on Electronic Computers, Vol. EC10, Dec. 1961, 729-733.

13. Newell, A. and Tonge, F., "An Introduction to Information Processing Language V," Comm's of the A.C.M., Vol. 3, April, 1960, 205-211.

14. McCarthy, J., "Recursive functions of symbolic expressions and their computations by machine, Part 1," Comms of the A.C.M., April, 1960, 184-195.

15. "Fortran System for the Control Data 1604 Computer," Computer Division, Control Data Corporation, Pub. 087A.

16. Samelson, K., and Bauer, F. L., "Sequential Formula Translation," Comm's of the A.C.M., Vol. 3., No. 2, Feb. 1960, pp 76.

Appendix A

Flow Diagrams of the Major Routines of the USNPGS Class Scheduling Program.

Figures A1 through A12 are detailed diagrams of the subprograms required by the control program or other subprograms.

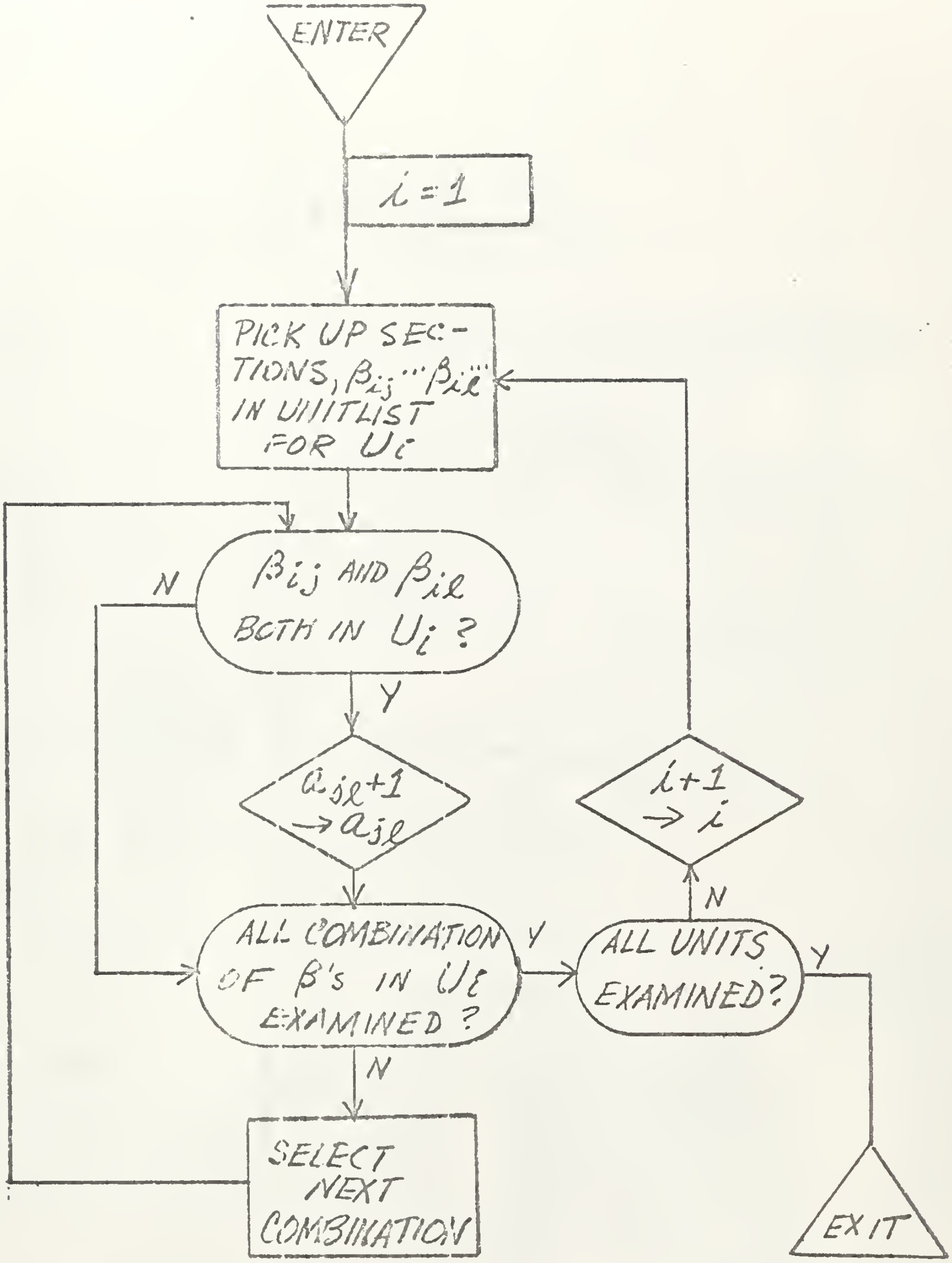

ENTER
$i=1$
PICK UP SECTIONS, $\beta_{ij} \cdots \beta_{il} \cdots$ IN UNITLIST FOR U_i
β_{ij} AND β_{il} BOTH IN U_i ?
N
Y
$a_{jl}+1 \rightarrow a_{jl}$
ALL COMBINATION OF β's IN U_i EXAMINED ?
Y
N
SELECT NEXT COMBINATION
ALL UNITS EXAMINED?
N
Y
$i+1 \rightarrow i$
EXIT

Figure A1
Form (A); β_j vs β_l

F_t^M = FLYQUOTA, PERIOD t

F_t^A = FLYERS ASSIGNED PERIOD t

F_i^N = FLYERS IN SECTION β_i

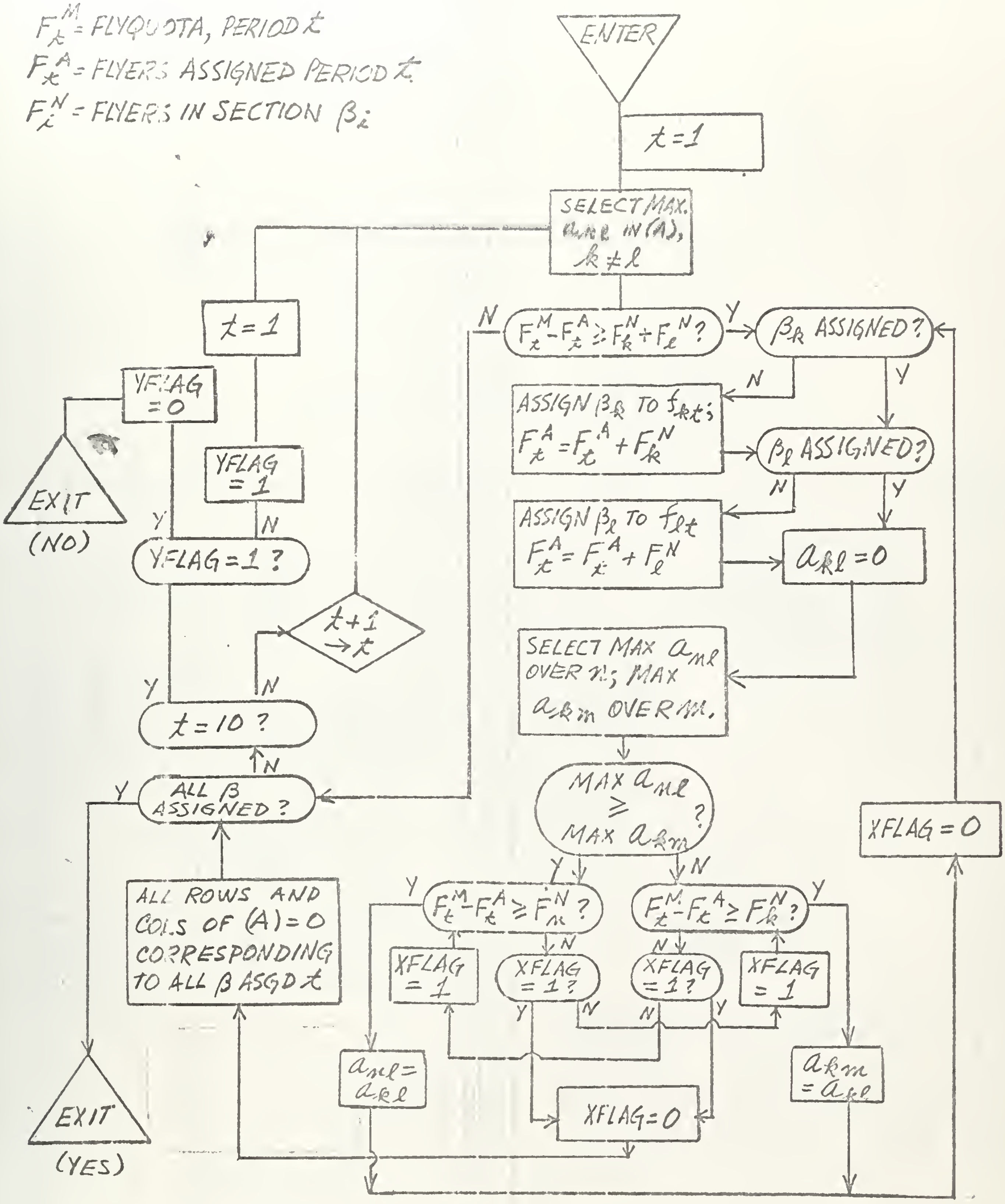

ENTER
$t=1$
SELECT MAX. a_{kl} IN (A), $k \neq l$
$F_t^M - F_t^A \geq F_k^N + F_l^N$?
N
Y
β_k ASSIGNED?
N
Y
ASSIGN β_k TO f_{kt}; $F_t^A = F_t^A + F_k^N$
β_l ASSIGNED?
N
Y
ASSIGN β_l TO f_{lt} $F_t^A = F_t^A + F_l^N$
$a_{kl} = 0$
SELECT MAX a_{nl} OVER n; MAX a_{km} OVER m.
MAX a_{nl} $\geq$ MAX a_{km} ?
Y
N
$F_t^M - F_t^A \geq F_n^N$?
$F_t^M - F_t^A \geq F_k^N$?
Y
N
XFLAG = 1
XFLAG = 1 ?
XFLAG = 1 ?
XFLAG = 1
Y
N
N
Y
$a_{nl} = a_{kl}$
XFLAG = 0
$a_{km} = a_{kl}$
XFLAG = 0
$t=1$
YFLAG = 0
YFLAG = 1
EXIT (NO)
Y
N
YFLAG = 1 ?
$t+1 \rightarrow t$
Y
N
$t = 10$?
N
Y
ALL β ASSIGNED?
ALL ROWS AND COLS OF (A) = 0 CORRESPONDING TO ALL β ASGD t
EXIT (YES)

Figure A2

To Form (F)

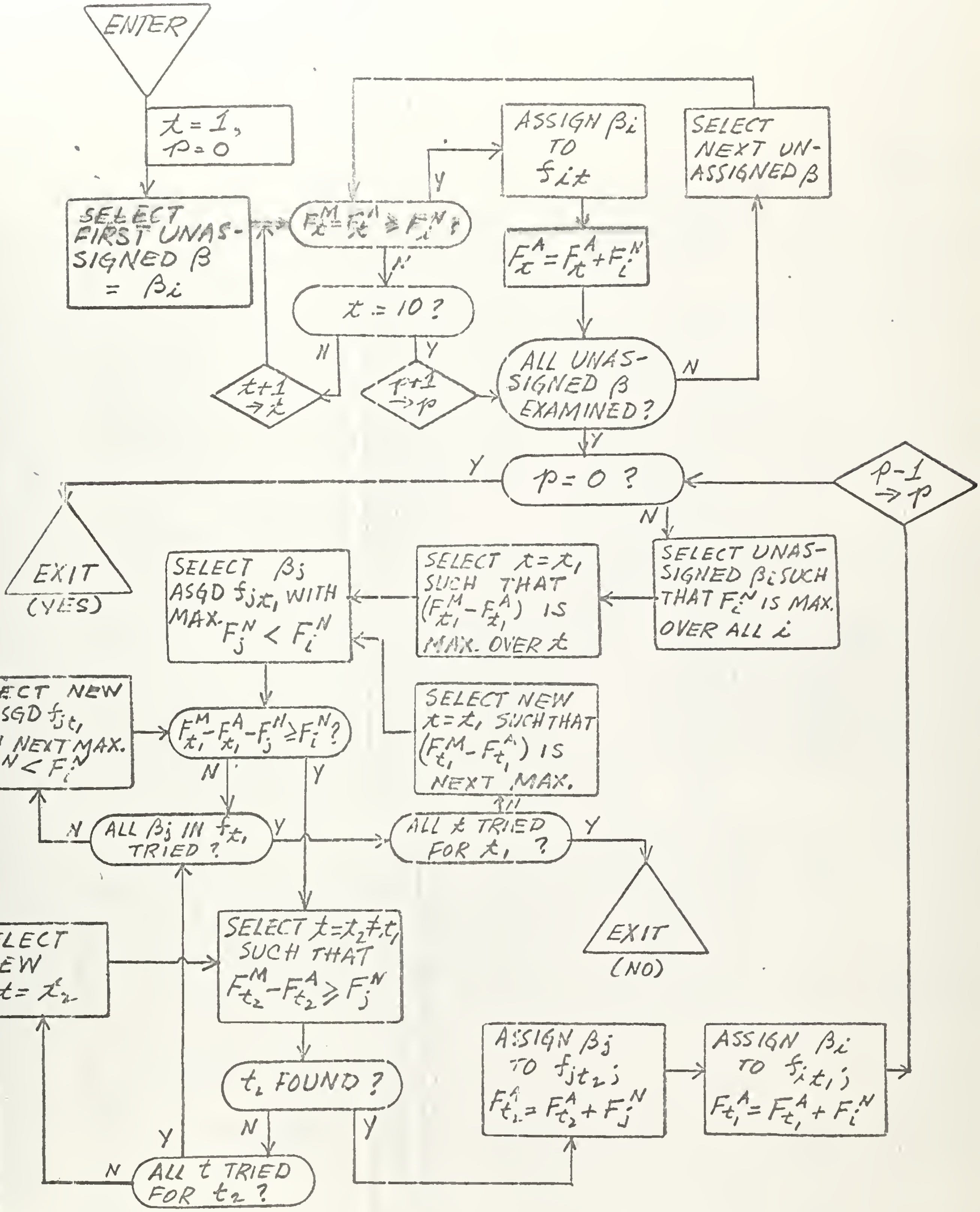

ENTER
$t = 1$, $p = 0$
SELECT FIRST UNASSIGNED $\beta = \beta_i$
$F_t^M - F_t^A \geq F_i^N$?
Y
N
ASSIGN β_i TO f_{it}
$F_t^A = F_t^A + F_i^N$
SELECT NEXT UNASSIGNED β
$t = 10$?
N
Y
$t+1 \to t$
$p+1 \to p$
ALL UNASSIGNED β EXAMINED?
N
Y
$p = 0$?
Y
N
$p-1 \to p$
EXIT (YES)
SELECT UNASSIGNED β_i SUCH THAT F_i^N IS MAX. OVER ALL i
SELECT $t = t_1$ SUCH THAT $(F_{t_1}^M - F_{t_1}^A)$ IS MAX. OVER t
SELECT β_j ASGD f_{jt_1} WITH MAX. $F_j^N < F_i^N$
SELECT NEW β_j ASGD f_{jt_1} WITH NEXT MAX. $F_j^N < F_i^N$
$F_{t_1}^M - F_{t_1}^A - F_j^N \geq F_i^N$?
N
Y
SELECT NEW $t = t_1$ SUCH THAT $(F_{t_1}^M - F_{t_1}^A)$ IS NEXT MAX.
N
ALL β_j IN f_{t_1} TRIED?
N
Y
ALL t TRIED FOR t_1?
Y
EXIT (NO)
SELECT NEW $t = t_2$
SELECT $t = t_2 \neq t_1$ SUCH THAT $F_{t_2}^M - F_{t_2}^A \geq F_j^N$
t_2 FOUND?
N
Y
ALL t TRIED FOR t_2?
Y
N
ASSIGN β_j TO f_{jt_2}; $F_{t_2}^A = F_{t_2}^A + F_j^N$
ASSIGN β_i TO f_{it_1}; $F_{t_1}^A = F_{t_1}^A + F_i^N$

Figure A3
Routine "Adjust Fly"

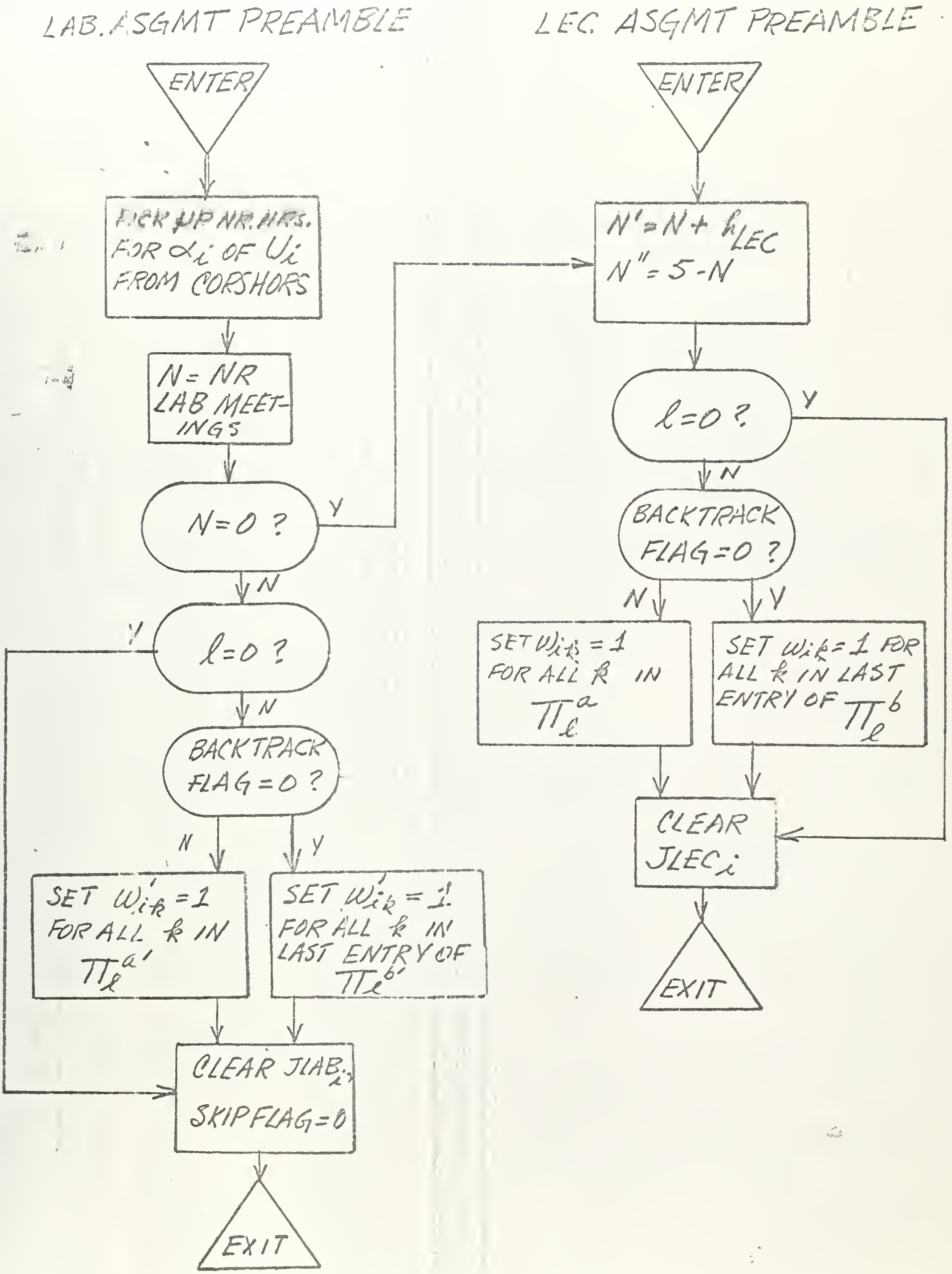

LAB. ASGMT PREAMBLE
ENTER
PICK UP NR. HRS. FOR α_i OF U_i FROM CORSHORS
N = NR LAB MEETINGS
N = 0 ?
Y
N
$\ell = 0$?
Y
N
BACKTRACK FLAG = 0 ?
N
Y
SET $W'_{ik} = 1$ FOR ALL k IN $\Pi_\ell^{a'}$
SET $W'_{ik} = 1$ FOR ALL k IN LAST ENTRY OF $\Pi_\ell^{b'}$
CLEAR $JLAB_i$, SKIPFLAG = 0
EXIT
LEC. ASGMT PREAMBLE
ENTER
$N' = N + h_{LEC}$
$N'' = 5 - N$
$\ell = 0$?
Y
N
BACKTRACK FLAG = 0 ?
N
Y
SET $W_{ik} = 1$ FOR ALL k IN Π_ℓ^a
SET $W_{ik} = 1$ FOR ALL k IN LAST ENTRY OF Π_ℓ^b
CLEAR $JLEC_i$
EXIT

Figure A4
Lecture and Laboratory
Preambles

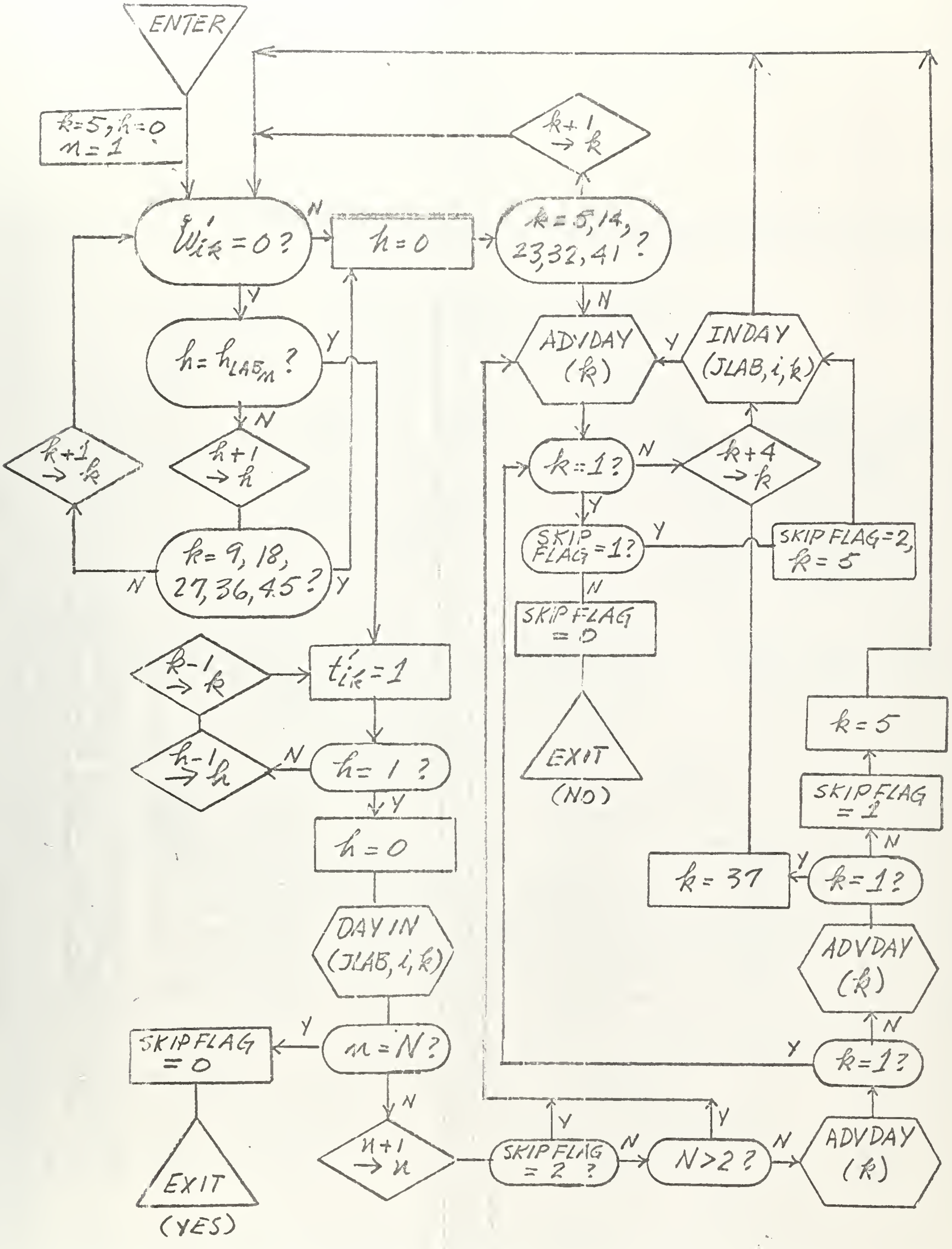
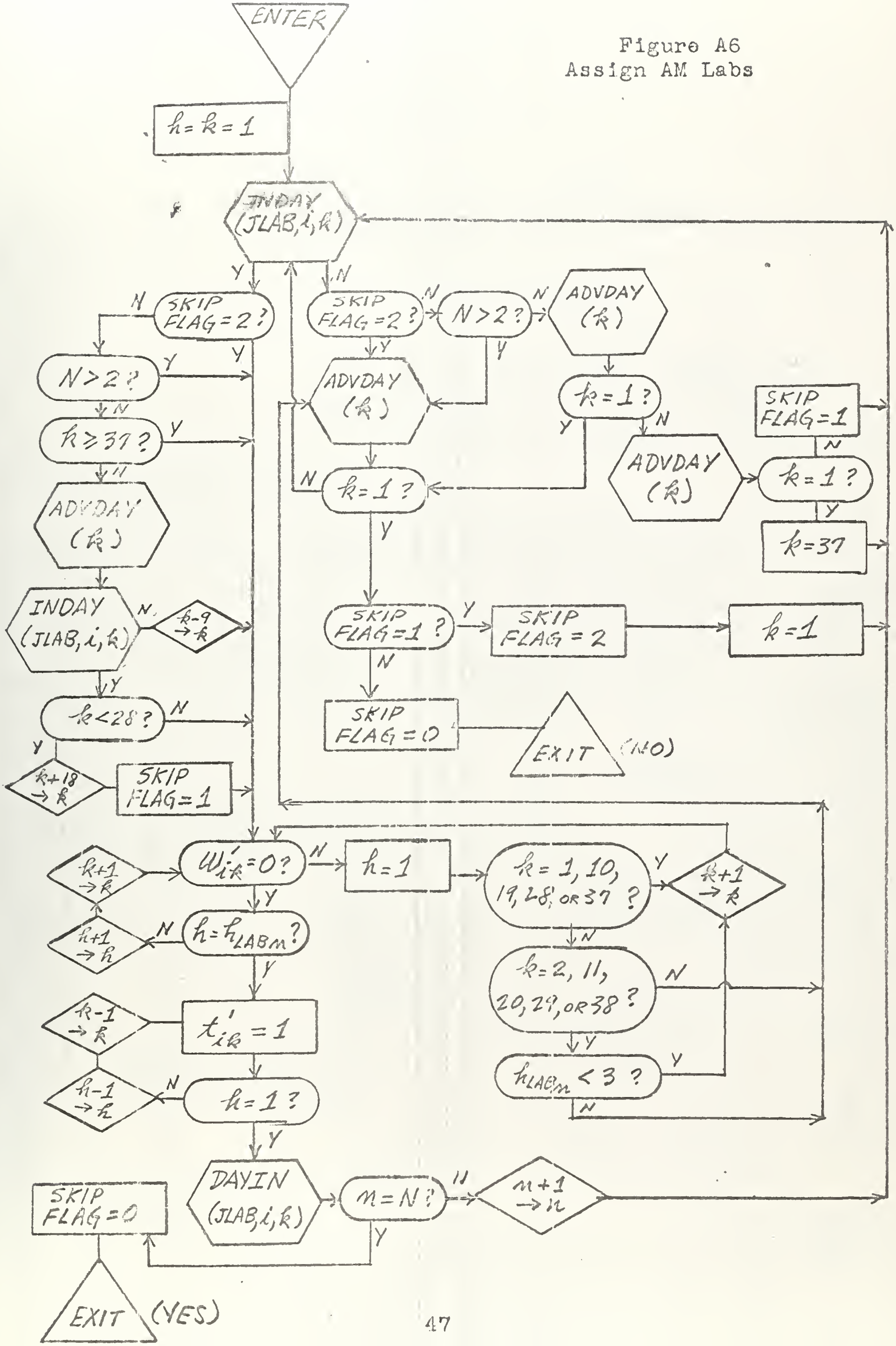


Figure A5
Assign PM Labs

Figure A6
Assign AM Labs



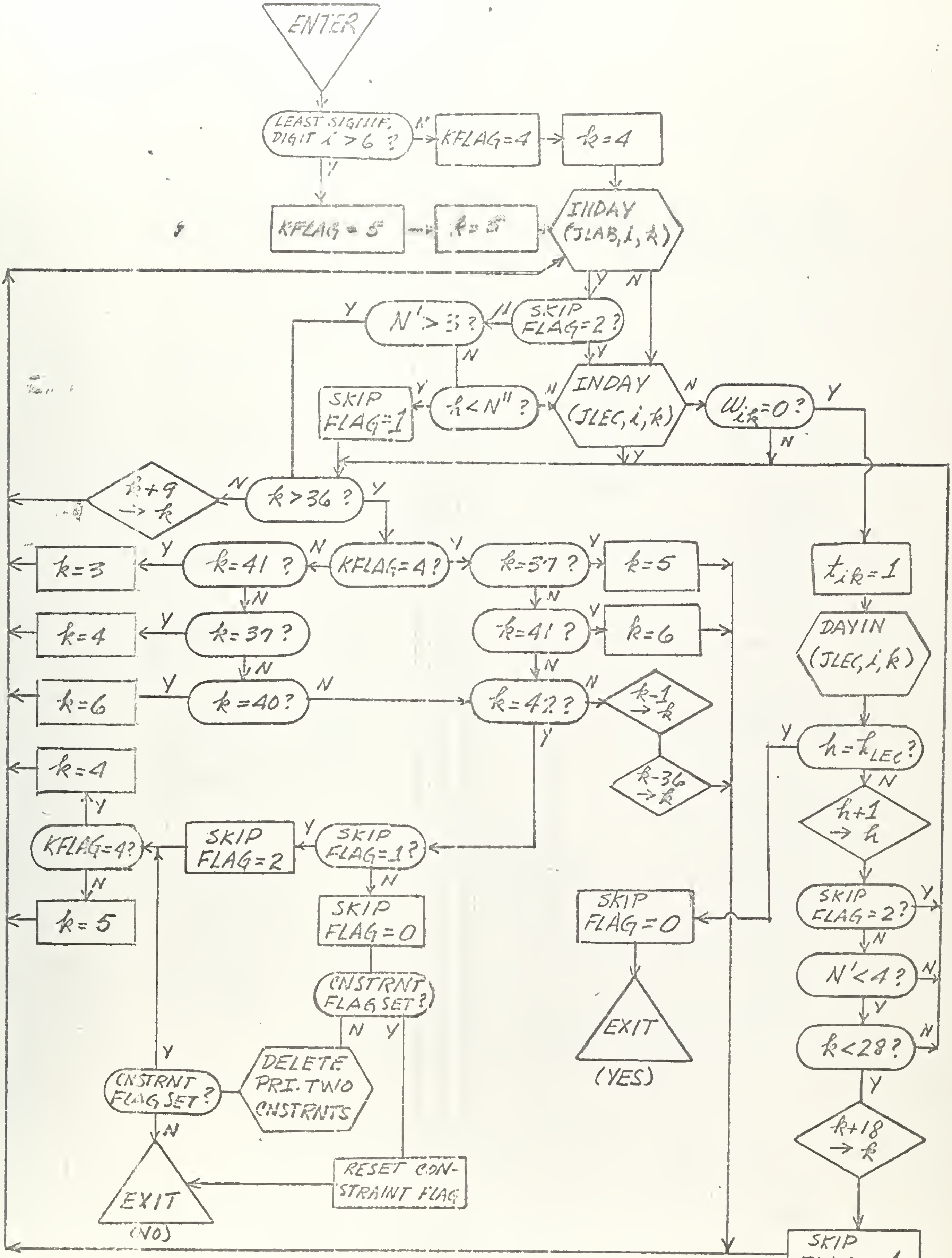


Figure A7
Assign Lectures

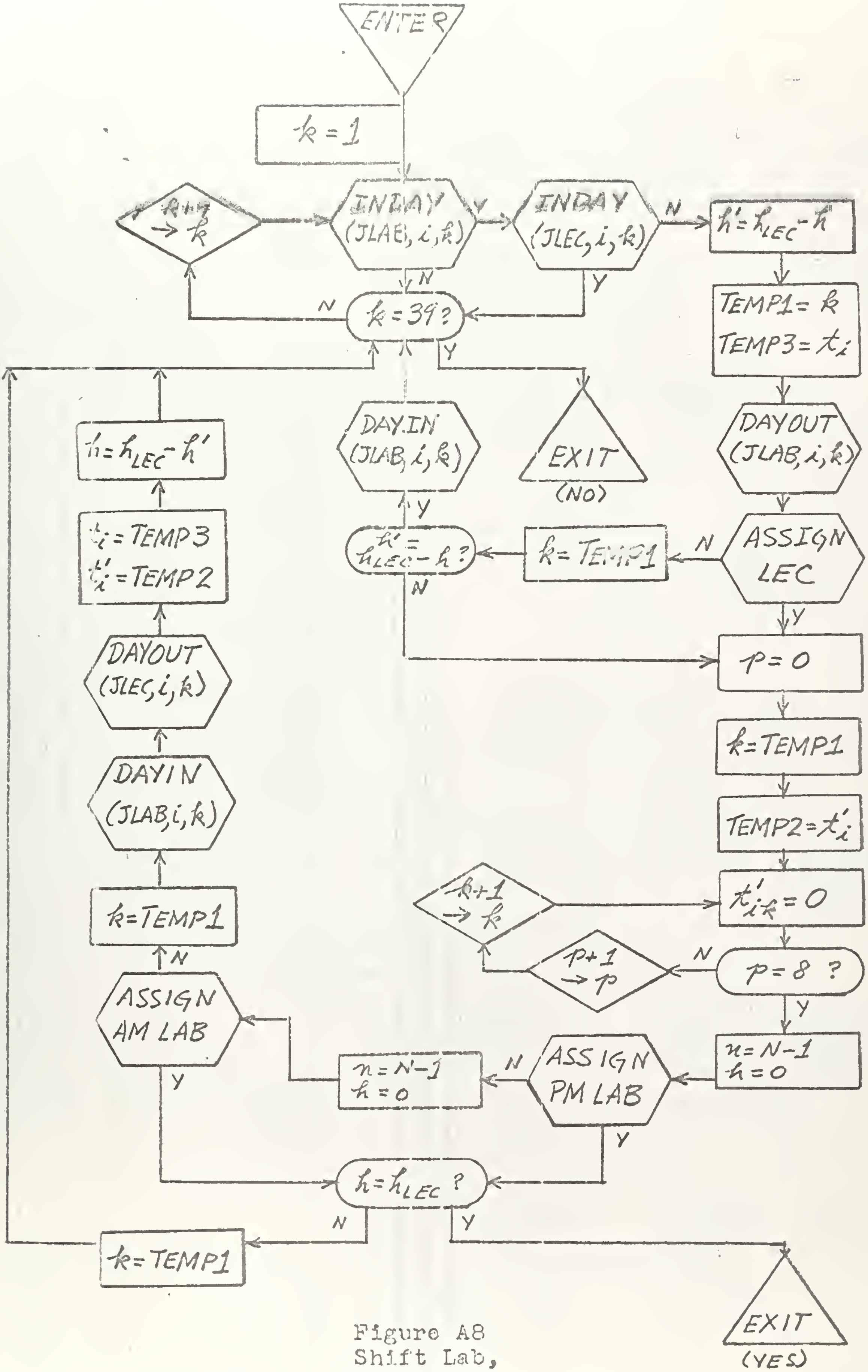
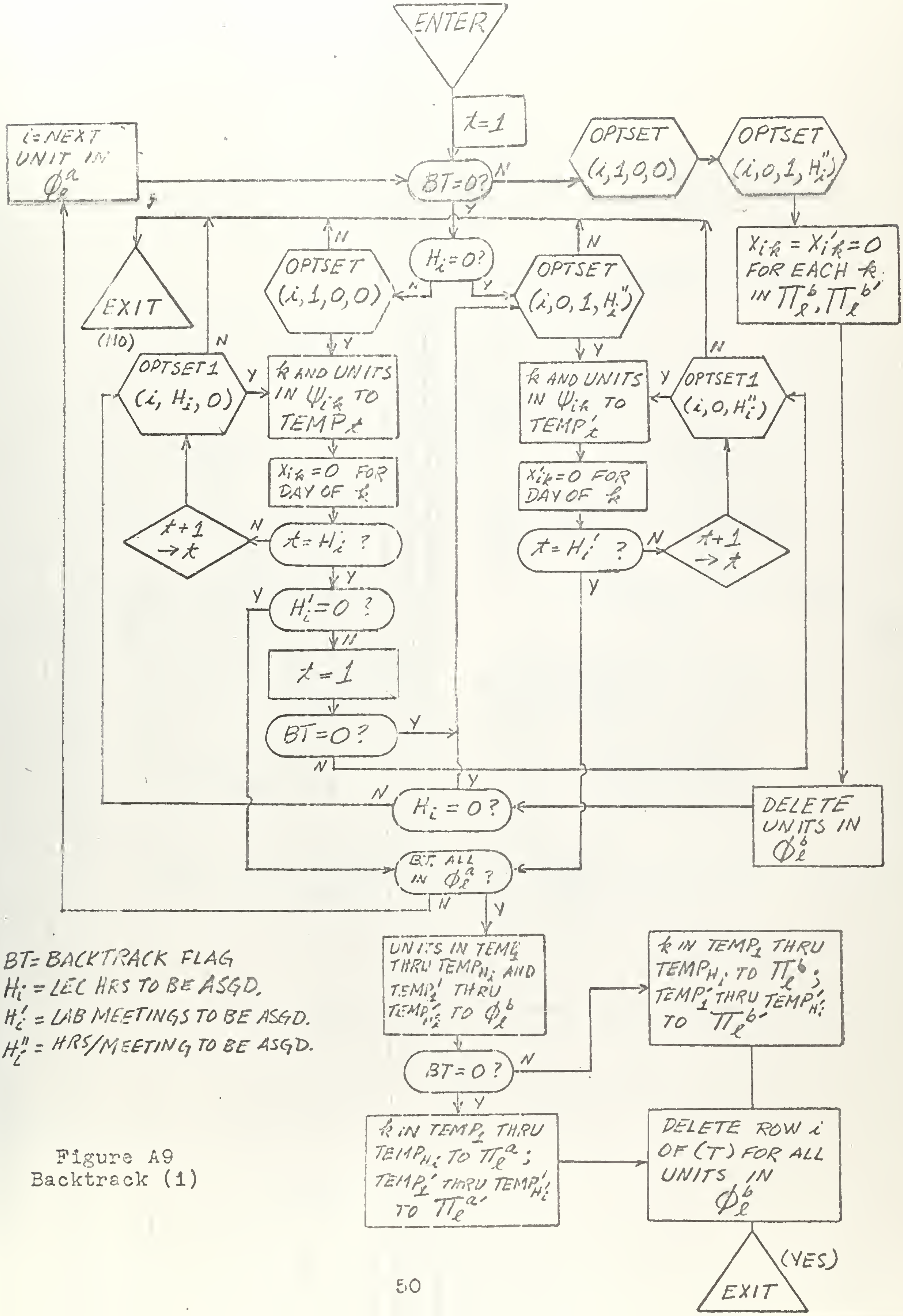


Figure A8
Shift Lab,
Assign Lec.


ENTER
$t=1$
i=NEXT UNIT IN ϕ_ℓ^a
$BT=0?$
N
Y
OPTSET $(i,1,0,0)$
OPTSET $(i,0,1,H_i'')$
$X_{ik}=X_{ik}'=0$ FOR EACH k IN $\Pi_\ell^b, \Pi_\ell^{b'}$
$H_i=0?$
EXIT
(NO)
OPTSET1 $(i, H_i, 0)$
k AND UNITS IN Ψ_{ik} TO $TEMP_t$
$X_{ik}=0$ FOR DAY OF k
$t+1 \rightarrow t$
$t=H_i$?
$H_i'=0$?
$t=1$
$BT=0?$
k AND UNITS IN Ψ_{ik} TO $TEMP_t'$
$X_{ik}'=0$ FOR DAY OF k
$t=H_i'$?
$t+1 \rightarrow t$
OPTSET1 $(i,0,H_i'')$
$H_i=0?$
DELETE UNITS IN ϕ_ℓ^b
B.T. ALL IN ϕ_ℓ^a ?
UNITS IN $TEMP_1$ THRU $TEMP_{H_i}$ AND $TEMP_1'$ THRU $TEMP_{H_i}'$ TO ϕ_ℓ^b
$BT=0?$
k IN $TEMP_1$ THRU $TEMP_{H_i}$ TO Π_ℓ^b; $TEMP_1'$ THRU $TEMP_{H_i}'$ TO $\Pi_\ell^{b'}$
k IN $TEMP_1$ THRU $TEMP_{H_i}$ TO Π_ℓ^a; $TEMP_1'$ THRU $TEMP_{H_i}'$ TO $\Pi_\ell^{a'}$
DELETE ROW i OF (T) FOR ALL UNITS IN ϕ_ℓ^b
(YES)
EXIT

BT= BACKTRACK FLAG
H_i = LEC HRS TO BE ASGD.
H_i' = LAB MEETINGS TO BE ASGD.
H_i'' = HRS/MEETING TO BE ASGD.

Figure A9
Backtrack (1)

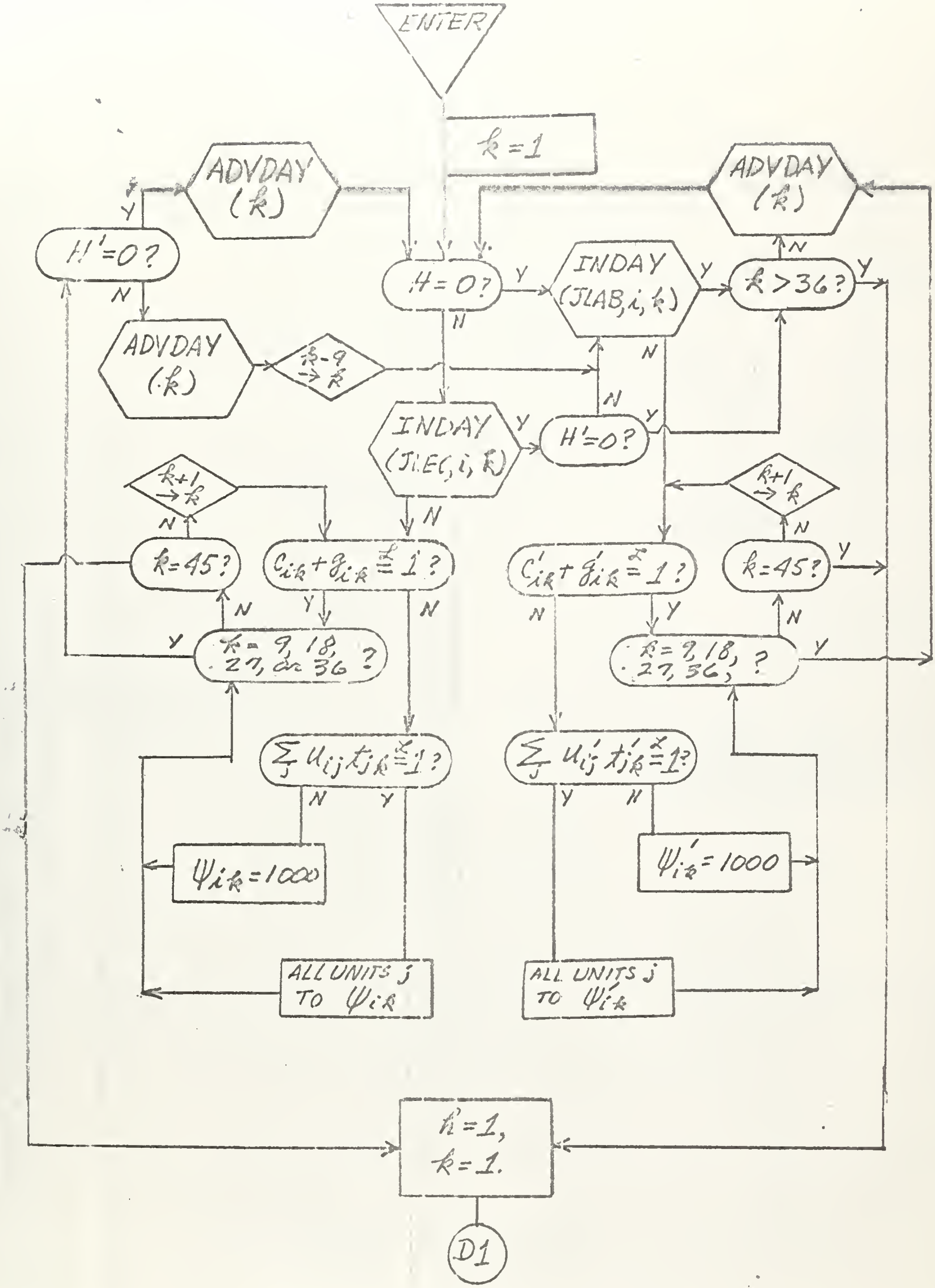


Figure A10
OPTSET (i,H,H',H")

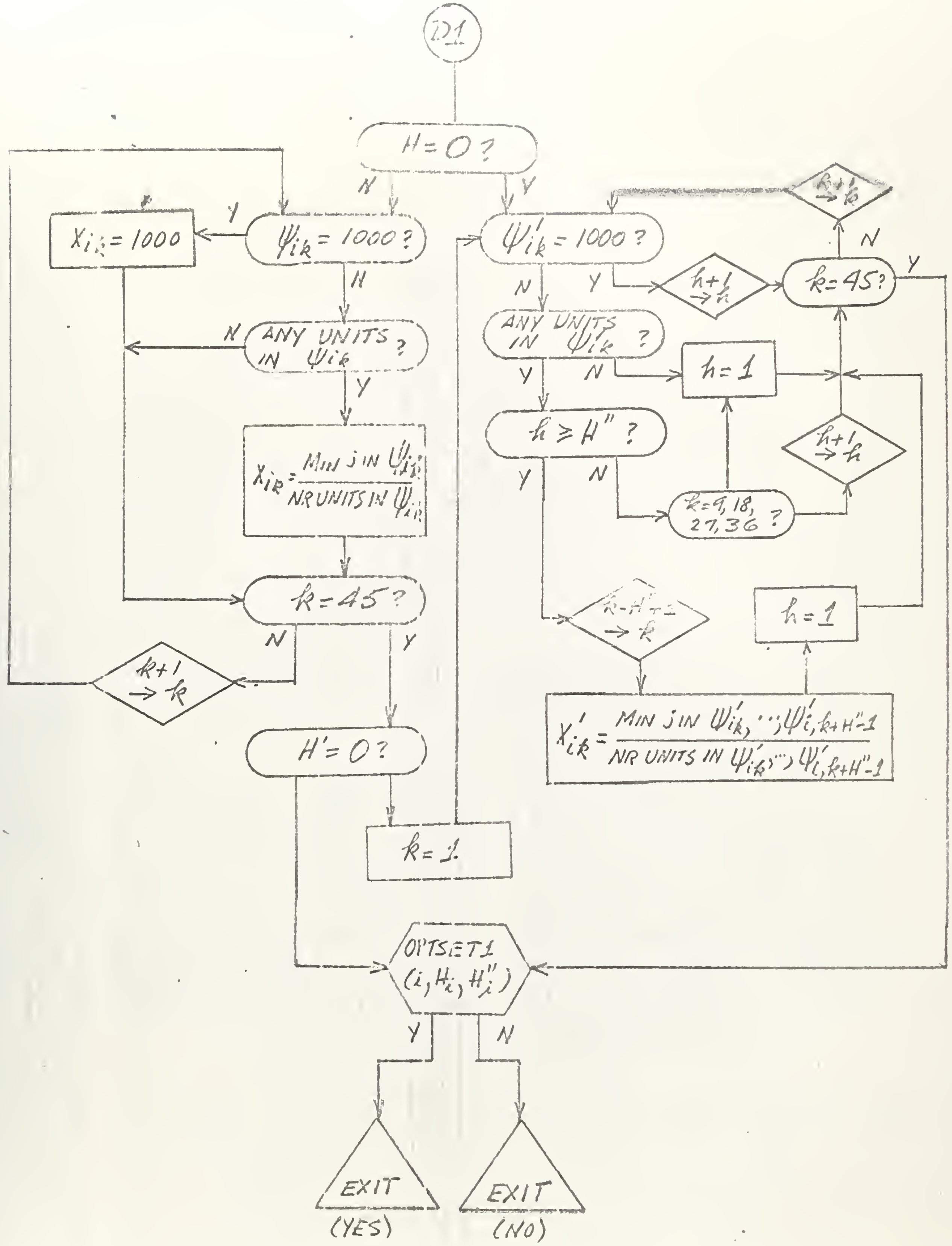

D1
H=0?
N
Y
Y
$X_{ik}=1000$
$\psi_{ik}=1000$?
N
N
ANY UNITS IN ψ_{ik} ?
Y
$X_{ik}=\frac{\text{MIN } j \text{ IN } \psi_{ik}}{\text{NR UNITS IN } \psi_{ik}}$
$k=45$?
N
Y
$k+1 \rightarrow k$
$H'=0$?
$k=1$
$\psi'_{ik}=1000$?
N
Y
$h+1 \rightarrow h$
$k=45$?
N
Y
$k+1 \rightarrow k$
ANY UNITS IN ψ'_{ik} ?
Y
N
$h=1$
$h \geq H''$?
Y
N
$k=9,18,27,36$?
$h+1 \rightarrow h$
$k-H''+1 \rightarrow k$
$h=1$
$X'_{ik}=\frac{\text{MIN } j \text{ IN } \psi'_{ik},\cdots,\psi'_{i,k+H''-1}}{\text{NR UNITS IN } \psi'_{ik},\cdots,\psi'_{i,k+H''-1}}$
OPTSET1 (i, H_i, H''_i)
Y
N
EXIT
(YES)
EXIT
(NO)

Figure A10
(cont'd)

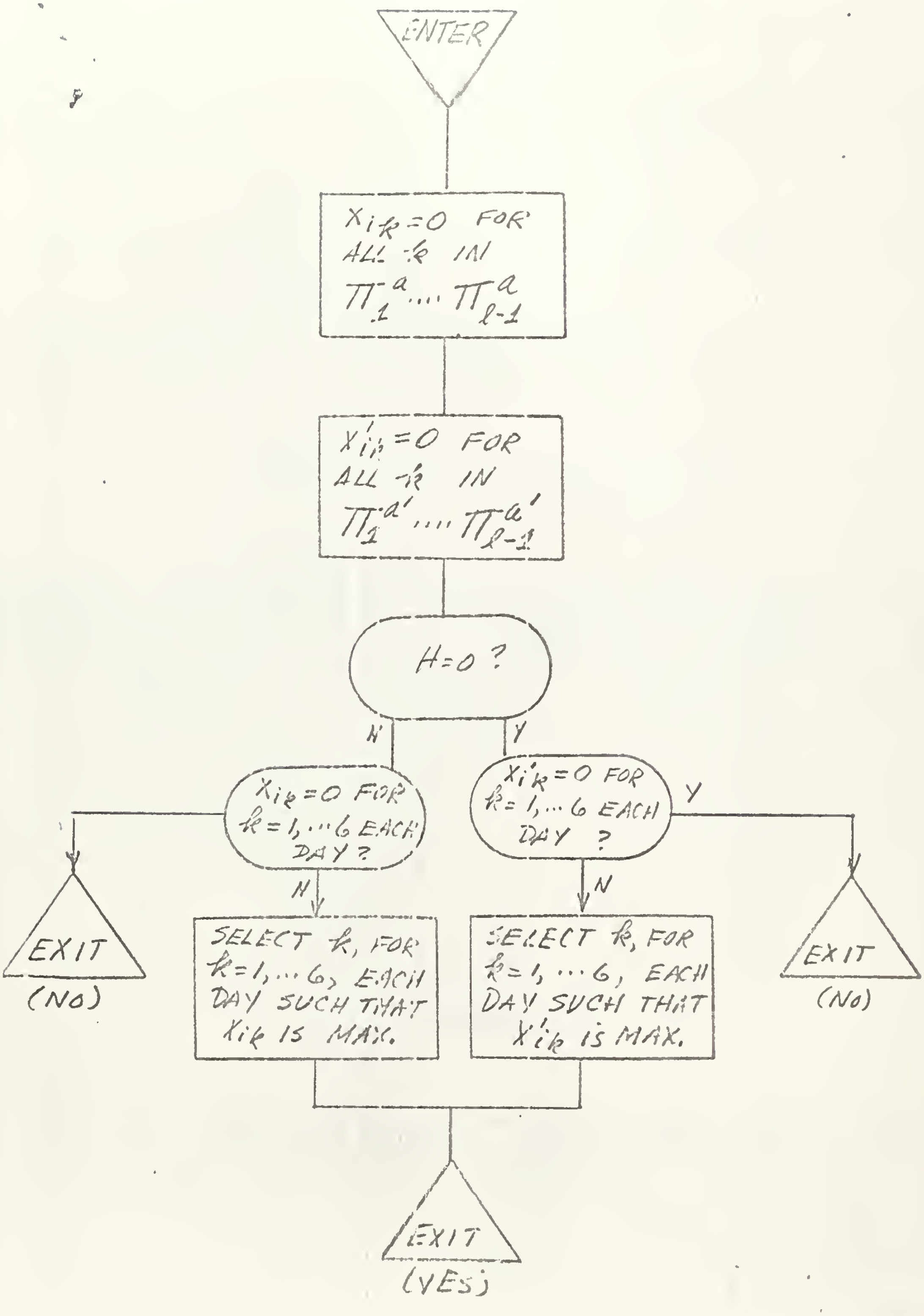

ENTER
$X_{ik}=0$ FOR ALL k IN $\Pi_1^a \dots \Pi_{\ell-1}^a$
$X'_{ik}=0$ FOR ALL k IN $\Pi_1^{a'} \dots \Pi_{\ell-1}^{a'}$
H=0 ?
N
Y
$X_{ik}=0$ FOR $k=1,\dots 6$ EACH DAY ?
$X'_{ik}=0$ FOR $k=1,\dots 6$ EACH DAY ?
Y
N
N
EXIT (NO)
SELECT k, FOR $k=1,\dots 6$, EACH DAY SUCH THAT X_{ik} IS MAX.
SELECT k, FOR $k=1,\dots 6$, EACH DAY SUCH THAT X'_{ik} IS MAX.
EXIT (NO)
EXIT (YES)

Figure All
OPTSET1 (i,H,H")

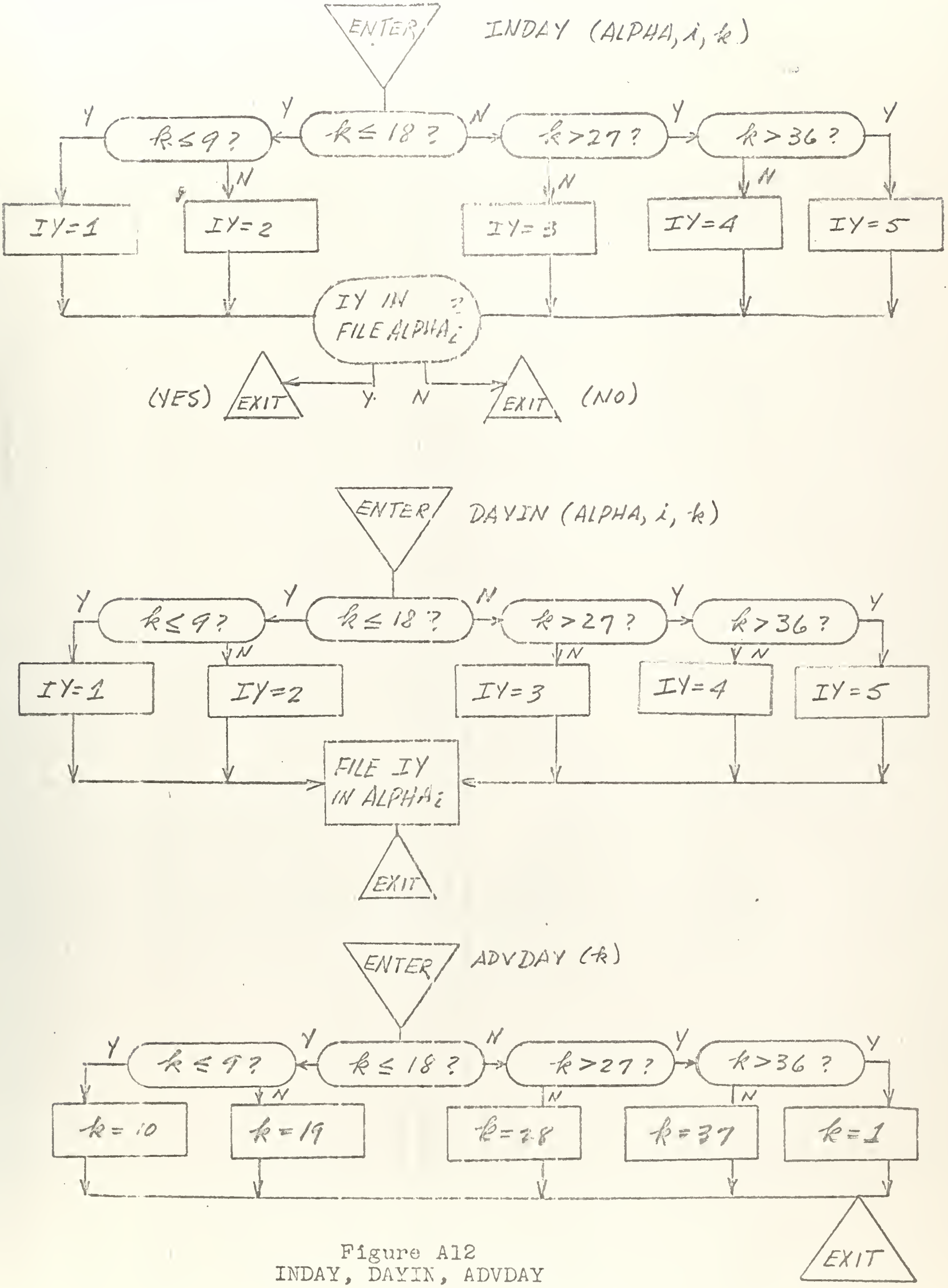


Figure A12
INDAY, DAYIN, ADVDAY